BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: PCT/DE 03/00942

Anmeldetag: 21. März 2003

Anmelder/Inhaber: PACT XPP Technologies AG, München/DE;
Martin V o r b a c h , München/DE.

Bezeichnung: Verfahren und Vorrichtung zur Datenverarbeitung

Priorität: 21.03.2002 DE 102 12 622.4; 21.03.2002 DE 102 12 621.6; 02.05.2002 DE 102 19 681.8; 02.05.2002 EP 02 009 868.7; 12.06.2002 DE 102 26 186.5; 20.06.2002 DE 102 27 650.1; 20.06.2002 EP PCT/EP 02/06865; 07.08.2002 DE 102 36 271.8; 07.08.2002 DE 102 36 272.6; 07.08.2002 DE 102 36 269.6; 16.08.2002 EP PCT/EP 02/10065; 21.08.2002 DE 102 38 174.7; 21.08.2002 DE 102 38 173.9; 21.08.2002 DE 102 38 172.0; 27.08.2002 DE 102 40 022.9; 27.08.2002 DE 102 40 000.8; 03.09.2002 DE PCT/DE 02/03278; 06.09.2002 DE 102 41 812.8; 18.09.2002 EP PCT/EP 02/10479; 18.09.2002 EP PCT/EP 02/10464; 19.09.2002 EP PCT/EP 02/10572; 10.10.2002 EP 02 022 692.4; 06.12.2002 EP 02 027 277.9; 07.01.2003 DE 103 00 380.0; 20.01.2003 EP PCT/EP 03/00624; 20.01.2003 DE PCT/DE 03/00152; 18.02.2003 DE PCT/DE 03/00489

IPC: noch nicht festgelegt

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 19. November 2003
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

A 9161
06/00
EDV-L

PCT

ANTRAG

Der Unterzeichnete beantragt, daß die vorliegende internationale Anmeldung nach dem Vertrag über die internationale Zusammenarbeit auf dem Gebiet des Patentwesens behandelt wird.

Vom An...

PCT/DE 03/00942
Internationales Aktenzeichen

21. März 2003 (21.03.03)
Internationales Anmeldedatum

RO/DE Deutsches Patent- und Markenamt
(German Patent and Trade Mark Office)
Name des Anmeldeamts und "PCT International Application"

Aktenzeichen des Anmelders oder Anwalts *(falls gewünscht)* *(max. 12 Zeichen)*: PACT29/PCTD

Feld Nr. I BEZEICHNUNG DER ERFINDUNG

Verfahren und Vorrichtung zur Datenverarbeitung

Feld Nr. II ANMELDER ☐ Diese Person ist gleichzeitig Erfinder

Name und Anschrift: *(Familienname, Vorname; bei juristischen Personen vollständige amtliche Bezeichnung. Bei der Anschrift sind die Postleitzahl und der Name des Staats anzugeben. Der in diesem Feld in der Anschrift angegebene Staat ist der Staat des Sitzes oder Wohnsitzes des Anmelders, sofern nachstehend kein Staat des Sitzes oder Wohnsitzes angegeben ist.)*

PACT XPP Technologies AG
Muthmannstraße 1
D-80939 München DE

Telefonnr.:

Telefaxnr.:

Fernschreibnr.:

Registrierungsnr. des Anmelders beim Amt:

Staatsangehörigkeit *(Staat)*: DE

Sitz oder Wohnsitz *(Staat)*: DE

Diese Person ist Anmelder für folgende Staaten: ☐ alle Bestimmungsstaaten ☒ alle Bestimmungsstaaten mit Ausnahme der Vereinigten Staaten von Amerika ☐ nur die Vereinigten Staaten von Amerika ☐ die im Zusatzfeld angegebenen Staaten

Feld Nr. III WEITERE ANMELDER UND/ODER (WEITERE) ERFINDER

Name und Anschrift: *(Familienname, Vorname; bei juristischen Personen vollständige amtliche Bezeichnung. Bei der Anschrift sind die Postleitzahl und der Name des Staats anzugeben. Der in diesem Feld in der Anschrift angegebene Staat ist der Staat des Sitzes oder Wohnsitzes des Anmelders, sofern nachstehend kein Staat des Sitzes oder Wohnsitzes angegeben ist.)*

VORBACH Martin
Gotthardstraße 117 a
D-80689 München
DE

Diese Person ist:
☐ nur Anmelder
☒ Anmelder und Erfinder
☐ nur Erfinder *(Wird dieses Kästchen angekreuzt, so sind die nachstehenden Angaben nicht nötig.)*

Registrierungsnr. des Anmelders beim Amt:

Staatsangehörigkeit *(Staat)*: DE

Sitz oder Wohnsitz *(Staat)*: DE

Diese Person ist Anmelder für folgende Staaten: ☐ alle Bestimmungsstaaten ☐ alle Bestimmungsstaaten mit Ausnahme der Vereinigten Staaten von Amerika ☒ nur die Vereinigten Staaten von Amerika ☐ die im Zusatzfeld angegebenen Staaten

☐ Weitere Anmelder und/oder (weitere) Erfinder sind auf einem Fortsetzungsblatt angegeben.

Feld Nr. IV ANWALT ODER GEMEINSAMER VERTRETER; ODER ZUSTELLANSCHRIFT

Die folgende Person wird hiermit bestellt/ist bestellt worden, um für den (die) Anmelder vor den zuständigen internationalen Behörden in folgender Eigenschaft zu handeln als: ☒ Anwalt ☐ gemeinsamer Vertreter

Name und Anschrift: *(Familienname, Vorname; bei juristischen Personen vollständige amtliche Bezeichnung. Bei der Anschrift sind die Postleitzahl und der Name des Staats anzugeben.)*

PIETRUK Claus Peter
Patentanwalt
Heinrich-Lilienfein-Weg 5
D-76229 Karlsruhe DE

Telefonnr.: ++49 721 462034

Telefaxnr.: ++49 721 469308

Fernschreibnr.:

Registrierungsnr. des Anwalts beim Amt:

☐ **Zustellanschrift:** Dieses Kästchen ist anzukreuzen, wenn kein Anwalt oder gemeinsamer Vertreter bestellt ist und statt dessen im obigen Feld eine spezielle Zustellanschrift angegeben ist.

Feld Nr. V BESTIMMUNG VON STAATEN *Bitte die entsprechenden Kästchen ankreuzen; wenigstens ein Kästchen muß angekreuzt werden.*

Die folgenden Bestimmungen nach Regel 4.9 Absatz a werden hiermit vorgenommen:

Regionales Patent

☒ **AP** **ARIPO-Patent:** GH Ghana, GM Gambia, KE Kenia, LS Lesotho, MW Malawi, MZ Mosambik, SD Sudan, SL Sierra Leone, SZ Swasiland, TZ Vereinigte Republik Tansania, UG Uganda, ZM Sambia, ZW Simbabwe und jeder weitere Staat, der Vertragsstaat des Harare-Protokolls und des PCT ist *(falls eine andere Schutzrechtsart oder ein sonstiges Verfahren gewünscht wird, bitte auf der gepunkteten Linie angeben)* und Gbm

☒ **EA** **Eurasisches Patent:** AM Armenien, AZ Aserbaidschan, BY Belarus, KG Kirgisistan, KZ Kasachstan, MD Republik Moldau, RU Russische Föderation, TJ Tadschikistan, TM Turkmenistan und jeder weitere Staat, der Vertragsstaat des Eurasischen Patentübereinkommens und des PCT ist

☒ **EP** **Europäisches Patent:** AT Österreich, BE Belgien, BG Bulgarien, CH &LI Schweiz und Liechtenstein, CY Zypern, CZ Tschechische Republik, DE Deutschland, DK Dänemark, EE Estland, ES Spanien, FI Finnland, FR Frankreich, GB Vereinigtes Königreich, GR Griechenland, IE Irland, IT Italien, LU Luxemburg, MC Monaco, NL Niederlande, PT Portugal, SE Schweden, SK Slowakei, TR Türkei und jeder weitere Staat, der Vertragsstaat des Europäischen Patentübereinkommens und des PCT ist

☒ **OA** **OAPI-Patent:** BF Burkina Faso, BJ Benin, CF Zentralafrikanische Republik, CG Kongo, CI Côte d'Ivoire, CM Kamerun, GA Gabun, GN Guinea, GQ Äquatorialguinea, GW Guinea-Bissau, ML Mali, MR Mauretanien, NE Niger, SN Senegal, TD Tschad, TG Togo und jeder weitere Staat, der Vertragsstaat der OAPI und des PCT ist *(falls eine andere Schutzrechtsart oder ein sonstiges Verfahren gewünscht wird, bitte auf der gepunkteten Linie angeben)*

Nationales Patent *(falls eine andere Schutzrechtsart oder ein sonstiges Verfahren gewünscht wird, bitte auf der gepunkteten Linie angeben):*

☒ AE Vereinigte Arabische Emirate
☒ AG Antigua und Barbuda
☒ AL Albanien
☒ AM Armenien
☒ AT Österreich und Gbm
☒ AU Australien
☒ AZ Aserbaidschan
☒ BA Bosnien-Herzegovina
☒ BB Barbados
☒ BG Bulgarien
☒ BR Brasilien
☒ BY Belarus
☒ BZ Belize
☒ CA Kanada
☒ CH & LI Schweiz und Liechtenstein
☒ CN China
☒ CO Kolumbien
☒ CR Costa Rica
☒ CU Kuba
☒ CZ Tschechische Republik und Gbm
☒ DE Deutschland und Gbm
☒ DK Dänemark und Gbm
☒ DM Dominica
☒ DZ Algerien
☒ EC Ecuador
☒ EE Estland und Gbm
☒ ES Spanien
☒ FI Finnland und Gbm
☒ GB Vereinigtes Königreich
☒ GD Grenada
☒ GE Georgien
☒ GH Ghana und ARIPO-Gbm
☒ GM Gambia und ARIPO-Gbm
☒ HR Kroatien
☒ HU Ungarn
☒ ID Indonesien
☒ IL Israel
☒ IN Indien
☒ IS Island
☒ JP Japan
☒ KE Kenia und ARIPO-Gbm
☒ KG Kirgisistan
☒ KP Demokratische Volksrepublik Korea
☒ KR Republik Korea
☒ KZ Kasachstan
☒ LC Saint Lucia
☒ LK Sri Lanka
☒ LR Liberia
☒ LS Lesotho und ARIPO-Gbm
☒ LT Litauen
☒ LU Luxemburg
☒ LV Lettland
☒ MA Marokko
☒ MD Republik Moldau
☒ MG Madagaskar
☒ MK Die ehemalige jugoslawische Republik Mazedonien
☒ MN Mongolei
☒ MW Malawi und ARIPO-Gbm
☒ MX Mexiko
☒ MZ Mosambik und ARIPO-Gbm
☒ NO Norwegen
☒ NZ Neuseeland
☒ OM Oman
☒ PH Philippinen
☒ PL Polen
☒ PT Portugal
☒ RO Rumänien
☒ RU Russische Föderation
☒ SD Sudan
☒ SE Schweden
☒ SG Singapur
☒ SI Slowenien
☒ SK Slowakei und ARIPO-Gbm
☒ SL Sierra Leone
☒ TJ Tadschikistan
☒ TM Turkmenistan
☒ TN Tunesien
☒ TR Türkei
☒ TT Trinidad und Tobago
☒ TZ Vereinigte Republik Tansania und ARIPO-Gbm
☒ UA Ukraine
☒ UG Uganda und ARIPO-Gbm
☒ US Vereinigte Staaten von Amerika
☒ UZ Usbekistan
☒ VN Vietnam
☒ YU Jugoslawien
☒ ZA Südafrika
☒ ZM Sambia und ARIPO-Gbm
☒ ZW Simbabwe und ARIPO-Gbm

sowie jeweils alle anderen zusätzlichen möglichen Schutzrechte in diesen Ländern

Kästchen für die Bestimmung von Staaten, die dem PCT nach der Veröffentlichung dieses Formblatts beigetreten sind.

☒ VC St. Vincent + Grenadine
☒ SC Seychellen
☒ SZ Swasiland u. ARIPO-Gbm
☒ NI Nicaragua
☐

Erklärung bzgl. vorsorglicher Bestimmungen: Zusätzlich zu den oben genannten Bestimmungen nimmt der Anmelder nach Regel 4.9 Absatz b auch alle anderen nach dem PCT zulässigen Bestimmungen vor mit Ausnahme der im Zusatzfeld genannten Bestimmungen, die von dieser Erklärung ausgenommen sind. Der Anmelder erklärt, daß diese zusätzlichen Bestimmungen unter dem Vorbehalt einer Bestätigung stehen und jede zusätzliche Bestimmung, die vor Ablauf von 15 Monaten ab dem Prioritätsdatum nicht bestätigt wurde, nach Ablauf dieser Frist als vom Anmelder zurückgenommen gilt. *(Die Bestätigung (einschließlich der Gebühren) muß beim Anmeldeamt innerhalb der Frist von 15 Monaten eingehen.)*

Feld Nr. VI PRIORITÄTSANSPRUCH

Die Priorität der folgenden früheren Anmeldung(en) wird hiermit in Anspruch genommen:

Anmeldedatum der früheren Anmeldung *(Tag/Monat/Jahr)*	Aktenzeichen der früheren Anmeldung	Ist die frühere Anmeldung eine:		
		nationale Anmeldung: Staat oder Mitglied der WTO	regionale Anmeldung:* regionales Amt	internationale Anmeldung: Anmeldeamt
Zeile (1) 21. März 2002 (21.03.02)	102 12 622.4	DE		
Zeile (2) 21. März 2002 (21.03.02)	102 12 621.6	DE		
Zeile (3) 02. Mai 2002 (02.05.02)	102 19 681.8	DE		
Zeile (4) 02. Mai 2002 (02.05.02)	02 009 868.7	EP		
Zeile (5) 12. Juni 2002 (12.06.02)	102 26 186.5	DE		

☐ Weitere Prioritätsansprüche sind im Zusatzfeld angegeben.

Das Anmeldeamt wird ersucht, eine beglaubigte Abschrift der oben bezeichneten früheren Anmeldung(en) zu erstellen und dem internationalen Büro zu übermitteln *(nur falls die frühere Anmeldung(en) bei dem Amt eingereicht worden ist (sind), das für die Zwecke dieser internationalen Anmeldung Anmeldeamt ist)*

☐ sämtliche Zeilen ☒ Zeile (1) ☒ Zeile (2) ☒ Zeile (3) ☐ Zeile (4) ☒ Zeile (5) ☒ weitere, siehe Zusatzfeld

** Falls es sich bei der früheren Anmeldung um eine ARIPO-Anmeldung handelt, geben Sie mindestens einen Staat an, der Mitgliedsstaat der Pariser Verbandsübereinkunft zum Schutz des gewerblichen Eigentums oder Mitglied der Welthandelsorganisation ist und für den oder das die frühere Anmeldung eingereicht wurde:*

Feld Nr. VII INTERNATIONALE RECHERCHENBEHÖRDE

Wahl der internationalen Recherchenbehörde (ISA) *(falls zwei oder mehr als zwei internationale Recherchenbehörden für die Ausführung der internationalen Recherche zuständig sind, geben Sie die von Ihnen gewählte Behörde an; der Zweibuchstaben-Code kann benutzt werden)*:

ISA /

Antrag auf Nutzung der Ergebnisse einer früheren Recherche; Bezugnahme auf diese frühere Recherche *(falls eine frühere Recherche bei der internationalen Recherchenbehörde beantragt oder von ihr durchgeführt worden ist)*:

Datum *(Tag/Monat/Jahr)* Aktenzeichen Staat *(oder regionales Amt)*

Feld Nr. VIII ERKLÄRUNGEN

Die Felder Nr. VIII (i) bis (v) enthalten die folgenden Erklärungen *(Kreuzen Sie unten die entsprechenden Kästchen an und geben Sie in der rechten Spalte für jede Erklärung deren Anzahl an)*		Anzahl der Erklärungen
☐ Feld Nr. VIII (i)	Erklärung hinsichtlich der Identität des Erfinders	
☐ Feld Nr. VIII (ii)	Erklärung hinsichtlich der Berechtigung des Anmelders, zum Zeitpunkt des internationalen Anmeldedatums, ein Patent zu beantragen und zu erhalten	
☒ Feld Nr. VIII (iii)	Erklärung hinsichtlich der Berechtigung des Anmelders, zum Zeitpunkt des internationalen Anmeldedatums, die Priorität einer früheren Anmeldung zu beanspruchen	1
☐ Feld Nr. VIII (iv)	Erfindererklärung (nur im Hinblick auf die Bestimmung der Vereinigten Staaten von Amerika)	
☐ Feld Nr. VIII (v)	Erklärung hinsichtlich unschädlicher Offenbarungen oder Ausnahmen von der Neuheitsschädlichkeit	

Zusatzfeld *Wird dieses Zusatzfeld nicht benutzt, so sollte dieses Blatt dem Antrag nicht beigefügt werden.*

1. *Wenn der **Platz in einem Feld nicht für alle Angaben ausreicht:** In diesem Fall schreiben Sie "Fortsetzung von Feld Nr. ..." [Nummer des Feldes angeben] und machen die Angaben entsprechend der in dem Feld, in dem der Platz nicht ausreicht, vorgeschriebenen Art und Weise, insbesondere:*

*(i) **Wenn mehr als zwei Anmelder und/oder Erfinder vorhanden sind und kein "Fortsetzungsblatt" zur Verfügung steht:** In diesem Fall schreiben Sie "Fortsetzung von Feld Nr. III" und machen für jede weitere Person die in Feld Nr. III vorgeschriebenen Angaben. Der in diesem Feld in der Anschrift angegebene Staat ist der Staat des Sitzes oder Wohnsitzes des Anmelders, sofern nachstehend kein Staat des Sitzes oder Wohnsitzes angegeben ist.*

*(ii) Wenn in Feld Nr. II oder III die Angabe **"die im Zusatzfeld angegebenen Staaten"** angekreuzt ist: In diesem Fall schreiben Sie "Fortsetzung von Feld Nr. II", "Fortsetzung von Feld Nr. III" bzw. "Fortsetzung von Feld Nr. II und Nr. III" und geben den Namen des Anmelders oder die Namen der Anmelder an und neben jedem Namen den Staat oder die Staaten (und/oder ggf. ARIPO-, eurasisches, europäisches oder OAPI-Patent), für die die bezeichnete Person Anmelder ist.*

*(iii) **Wenn der in Feld Nr. II oder III genannte Erfinder oder Erfinder/Anmelder nicht für alle Bestimmungsstaaten oder für die Vereinigten Staaten von Amerika als Erfinder benannt ist:** In diesem Fall schreiben Sie "Fortsetzung von Feld Nr. II", "Fortsetzung von Feld Nr. III" bzw. "Fortsetzung von Feld Nr. II und Nr. III" und geben den Namen des Erfinders oder die Namen der Erfinder an und neben jedem Namen den Staat oder die Staaten (und/oder ggf. ARIPO-, eurasisches, europäisches oder OAPI-Patent), für die die bezeichnete Person Erfinder ist.*

*(iv) Wenn zusätzlich zu dem Anwalt oder den Anwälten, die in Feld Nr. IV angegeben sind, **weitere Anwälte bestellt sind:** In diesem Fall schreiben Sie "Fortsetzung von Feld Nr. IV" und machen für jeden weiteren Anwalt die entsprechenden, in Feld Nr. IV vorgeschriebenen Angaben.*

*(v) Wenn in Feld Nr. V bei einem Staat (oder bei OAPI) die Angabe **"Zusatzpatent"** oder **"Zusatzzertifikat,"** oder wenn in Feld Nr. V bei den Vereinigten Staaten von Amerika die Angabe **"Fortsetzung"** oder **"Teilfortsetzung"** hinzugefügt wird: In diesem Fall schreiben Sie "Fortsetzung von Feld Nr. V" und geben den Namen des betreffenden Staats (oder OAPI) an und nach dem Namen jedes solchen Staats (oder OAPI) das Aktenzeichen des Hauptschutzrechts oder der Hauptschutzrechtsanmeldung und das Datum der Erteilung des Hauptschutzrechts oder der Einreichung der Hauptschutzrechtsanmeldung.*

*(vi) Wenn in Feld Nr. VI **die Priorität von mehr als fünf früheren Anmeldungen beansprucht wird** In diesem Fall schreiben Sie "Fortsetzung von Feld Nr. VI" und machen für jede weitere frühere Anmeldung die entsprechenden, in Feld Nr. VI vorgeschriebenen Angaben.*

2. *Wenn, im Hinblick auf die **Erklärung bzgl. vorsorglicher Bestimmungen** in Feld Nr. V, der Anmelder Staaten von dieser Erklärung ausnehmen möchte: In diesem Fall schreiben Sie "Bestimmung(en), die von der Erklärung bzgl. vorsorglicher Bestimmungen ausgenommen ist(sind)" und geben den Namen oder den Zweibuchstaben-Code jedes so ausgeschlossenen Staates an.*

Fortsetzung von Feld VI:

6) 20. Juni 2002 102 27 650.1 DE
7) 20. Juni 2002 PCT/EP 02/06865 EP hinterlegt mit Wirkung für TT
8) 07. Aug. 2002 102 36 271.8 DE
9) 07. Aug. 2002 102 36 272.6 DE
10) 07. Aug. 2002 102 36 269.6 DE
11) 16. Aug. 2002 PCT/EP 02/10065 EP hinterlegt mit Wirkung für TT
12) 21. Aug. 2002 102 38 174.7 DE
13) 21. Aug. 2002 102 38 173.9 DE
14) 21. Aug. 2002 102 38 172.0 DE
15) 27. Aug. 2002 102 40 022.9 DE
16) 27. Aug. 2002 102 40 000.8 DE
17) 03. Sept. 2002 PCT/DE 02/03278 DE hinterlegt mit Wirkung für TT
18) 06. Sept. 2002 102 41 812.8 DE → X
19) 18. Sept. 2002 PCT/EP 02/10479 EP hinterlegt mit Wirkung für TT
20) 18. Sept. 2002 PCT/EP 02/10464 EP hinterlegt mit Wirkung für TT
21) 19. Sept. 2002 PCT/EP 02/10572 EP hinterlegt mit Wirkung für TT
22) 10. Okt. 2002 02 022 692.4 EP ✓
23) 06. Dez. 2002 02 027 277.9 EP ✓
24) 07. Jan. 2003 103 00 380.0 DE
25) 20. Jan. 2003 PCT/EP 03/00624 EP hinterlegt mit Wirkung für TT
26) 20. Jan. 2003 PCT/DE 03/00152 DE hinterlegt mit Wirkung für TT
27) 18. Feb. 2003 PCT/DE 03/00489 DE hinterlegt mit Wirkung für TT

Abb. 30 nicht öffentlich

Feld Nr. VIII (iii) ERKLÄRUNG: BERECHTIGUNG, DIE PRIORITÄT EINER FRÜHEREN ANMELDUNG ZU BEANSPRUCHEN

Die Erklärung muß dem in Abschnitt 213 vorgeschriebenen Wortlaut entsprechen; siehe Anmerkungen zu den Feldern VIII, VIII (bis (v) (allgemein) und insbesondere die Anmerkungen zum Feld Nr. VIII (iii). Wird dieses Feld nicht benutzt, so sollte dieses Blatt in Antrag nicht beigefügt werden.

Erklärung hinsichtlich der Berechtigung des Anmelders, zum Zeitpunkt des internationalen Anmeldedatums, die Priorität der unten aufgeführten früheren Anmeldung zu beanspruchen, in Fällen, in denen der Anmelder nicht auch der Anmelder der früheren Anmeldung ist, oder in Fällen, in denen sich der Name des Anmelders seit der Einreichung der früheren Anmeldung geändert hat (Regeln 4.17iii und 51*bis*.1 Absatz a Ziffer iii):

Die PACT XPP Technologies AG ist kraft des nachfolgend Aufgeführten berechtigt, die Prioritäten der früheren aufgeführten Anmeldungen zu beanspruchen:

Der Name des Anmelders hat sich gemäß Änderung des Gesellschaftsvertrages vom 08. Juli 2002 von PACT Informationstechnologie GmbH gemäß formwechselnder Umwandlung in PACT XPP Technologies AG geändert; Datum der letzten Eintragung im Handelsregister 27. August 2002 - gemäß beiliegendem Handelsregisterauszug vom 12. September 2002.

☐ Diese Erklärung wird auf dem folgenden Blatt fortgeführt, "Fortsetzungsblatt für Feld Nr. VIII (iii)".

Feld Nr. IX KONTROLLISTE; EINREICHUNGSSPRACHE

Diese internationale Anmeldung **enthält:**

(a) die folgende Anzahl an **Blättern Papier:**

Antrag (inklusive Erklärungsblätter)	:	6
Beschreibung (ohne Sequenzprotokollteil)	:	27
Ansprüche	:	6
Zusammenfassung	:	1
Zeichnungen	:	7
Teilanzahl	:	47
Sequenzprotokollteil der Beschreibung *(Anzahl der Blätter, soweit auf Papier eingereicht wird, unabhängig davon, ob zusätzlich auch in computerlesbarer Form eingereicht wird)*	:	
Gesamtanzahl	:	

(b) Sequenzprotokollteil der Beschreibung in computerlesbarer Form eingereicht

(i) ☐ ausschließlich in dieser Form (nach Abschnitt 801(a)(i))

(ii) ☐ zusätzlich zur Einreichung auf Papier (nach Abschnitt 801(a)(ii))

Art und Anzahl der Datenträger (Diskette, CD-ROM, CD-R oder sonstige), auf denen der Sequenzprotokollteil enthalten ist *(zusätzlich eingereichte Kopien unter Punkt 9(ii) in der rechten Spalte angeben)*

Dieser internationalen Anmeldung liegen die folgenden Unterlagen bei *(kreuzen Sie die entsprechenden Kästchen an und geben Sie in der rechten Spalte jeweils die Anzahl der beiliegenden Exemplare an)* — Anzahl

1. ☐ Blatt für die Gebührenberechnung :
2. ☐ Original einer gesonderten Vollmacht :
3. ☐ Original einer allgemeinen Vollmacht :
4. ☐ Kopie der allgemeinen Vollmacht; Aktenzeichen (falls vorhanden): :
5. ☐ Begründung für das Fehlen einer Unterschrift :
6. ☐ Prioritätsbeleg(e), in Feld Nr. VI durch folgende Zeilennummer(n) gekennzeichnet: :
7. ☐ Übersetzung der internationalen Anmeldung in die folgende Sprache: :
8. ☐ Gesonderte Angaben zu hinterlegten Mikroorganismen oder anderem biologischen Material :
9. ☐ Sequenzprotokoll in computerlesbarer Form (geben Sie zusätzlich die Art und Anzahl der beiliegenden Datenträger an (Diskette, CD-ROM, CD-R oder sonstige))
   - (i) ☐ Kopie ausschließlich für die Zwecke der internationalen Recherche nach Regel 13*ter* (und nicht als Teil der internationalen Anmeldung) :
   - (ii) ☐ *(nur falls Feld (b)(i) oder (b)(ii) in der linken Spalte angekreuzt wurde)* zusätzliche Kopien einschließlich, soweit zutreffend, einer Kopie für die Zwecke der internationalen Recherche nach Regel 13*ter* :
   - (iii) ☐ zusammen mit entsprechender Erklärung, daß die Kopie(n) mit dem in der linken Spalte aufgeführten Sequenzprotokollteil identisch ist (sind) :
10. ☒ Sonstige *(einzeln aufführen)*: 2 Blatt HR-Auszug

Abbildung der Zeichnungen, die mit der Zusammenfassung veröffentlicht werden soll (Nr.):

Sprache, in der die internationale Anmeldung eingereicht wird: DE

Feld Nr. X UNTERSCHRIFT DES ANMELDERS, DES ANWALTS ODER DES GEMEINSAMEN VERTRETERS

Der Name jeder unterzeichnenden Person ist neben der Unterschrift zu wiederholen, und es ist anzugeben, sofern sich dies nicht eindeutig aus dem Antrag ergibt, in welcher Eigenschaft die Person unterzeichnet.

D-76229 Karlsruhe, den 21. März 2003

Kanzlei Pietruk
Heinrich-Lilienlein-Weg 5
D-76229 Karlsruhe

Vom Anmeldeamt auszufüllen

1. Datum des tatsächlichen Eingangs dieser internationalen Anmeldung: 21. März 2003 (21.03.03)

2. Zeichnungen:
   ☒ eingegangen:
   ☐ nicht eingegangen:

3. Geändertes Eingangsdatum aufgrund nachträglich, jedoch fristgerecht eingegangener Unterlagen oder Zeichnungen zur Vervollständigung dieser internationalen Anmeldung:

4. Datum des fristgerechten Eingangs der angeforderten Richtigstellungen nach Artikel 11(2) PCT:

5. Internationale Recherchenbehörde *(falls zwei oder mehr zuständig sind)*: ISA / EP

6. ☐ Übermittlung des Recherchenexemplars bis zur Zahlung der Recherchengebühr aufgeschoben

Vom Internationalen Büro auszufüllen

Datum des Eingangs des Aktenexemplars beim Internationalen Büro:

Beschreibung

Die vorliegende Erfindung befaßt sich mit der Integration und/oder engen Kopplung von rekonfigurierbaren Prozessoren mit Standardprozessoren, dem Datenaustausch und der Synchronisation der Datenverarbeitung sowie Compilern hierfür.

Unter einer rekonfigurierbaren Architektur werden vorliegend Bausteine (VPU) mit konfigurierbarer Funktion und/oder Vernetzung verstanden, insbesondere integrierte Bausteine mit einer Mehrzahl von ein- oder mehrdimensional angeordneten arithmetischen und/oder logischen und/oder analogen und/oder speichernden und/oder intern/extern vernetzenden Baugruppen, die direkt oder durch ein Bussystem miteinander verbunden sind.

Zur Gattung dieser Bausteine zählen insbesondere systolische Arrays, neuronale Netze, Mehrprozessor Systeme, Prozessoren mit mehreren Rechenwerken und/oder logischen Zellen und/oder kommunikativen/peripheren Zellen (IO), Vernetzungs- und Netzwerkbausteine wie z.B. Crossbar-Schalter, ebenso wie bekannte Bausteine der Gattung FPGA, DPGA, Chameleon, XPUTER, etc.. Hingewiesen wird insbesondere in diesem Zusammenhang auf die folgenden Schutzrechte und Anmeldungen desselben Anmelders:
P 44 16 881 A1, DE 197 81 412 A1, DE 197 81 483 A1, DE 196 54 846 A1, DE 196 54 593 A1, DE 197 04 044.6 A1, DE 198 80 129 A1, DE 198 61 088 A1, DE 199 80 312 A1, PCT/DE 00/01869, DE 100 36 627 A1, DE 100 28 397 A1, DE 101 10 530 A1, DE 101 11 014 A1, PCT/EP 00/10516, EP 01 102 674 A1, DE 198 80 128 A1, DE 101 39 170 A1, DE 198 09 640 A1, DE 199 26 538.0 A1, DE 100 50 442 A1, PCT/EP 02/02398, DE 102 40 000, DE 102 02 044, DE 102 02 175, DE 101 29 237, DE 101 42 904, DE 101 35 210, EP 01 129 923, PCT/EP 02/10084, DE 102 12 622, DE 102 36 271, DE 102 12 621, EP 02 009 868, DE 102 36 272, DE 102 41 812, DE 102 36 269,
DE 102 43 322, EP 02 022 692, DE 103 00 380, DE 103 10 195, sowie die EP 02 001 331 und EP 02 027 277. Diese sind hiermit zu Offenbarungszwecken vollumfänglich eingegliedert

Die o.g. Architektur wird beispielhaft zur Verdeutlichung herangezogen und im folgenden VPU genannt. Die Architektur besteht aus beliebigen, typisch grobgranularen arithmetischen, logischen (auch Speicher) und/oder Speicherzellen und/oder Vernetzungszellen und/oder kommunikativen/peripheren (IO) Zellen (PAEs), die zu einer ein- oder mehrdimensionalen Matrix (PA) angeordnet sein können, wobei die Matrix unterschiedliche beliebig ausgestaltete Zellen aufweisen kann, und auch die Bussysteme dabei als Zellen verstanden werden können. Der Matrix als ganzes oder Teilen davon zugeordnet ist eine Konfigurationseinheit (CT), die die Vernetzung und Funktion des PA durch Konfiguration bestimmt. Es kann eine feingranulare Steuerlogik vorgesehen sein.

Verschiedene Methoden zum Ankoppeln von rekonfigurierbaren Prozessoren an Standardprozessoren sind bekannt. Diese sehen für gewöhnlich eine lose Kopplung vor. Die Art und Weise der Kopplung bedarf in vielen Aspekten noch einer Weiterentwicklung, genauso wie die für die gemeinsame Abarbeitung von Programmen auf Kombinationen aus rekonfigurierbaren Prozessoren und Standardprozessoren vorgesehenen Kompilier- bzw. Betriebsverfahren.

Die Aufgabe der Erfindung ist es, Neues für die gewerbliche Nutzung bereitzustellen.

Die Lösung der Aufgabe wird unabhängig beansprucht. Bevorzugte Ausführungsformen befinden sich in den Unteransprüchen.

Beschreibung der Erfindung

Ein Standardprozessor, z.B. ein RISC, CISC, DSP (CPU), werde mit einem rekonfigurierbaren Prozessor (VPU) gekoppelt. Zwei unterschiedliche, bevorzugt jedoch zugleich implementierte und/oder implementierbare Kopplungsvarianten werden beschrieben.

Eine erste Variante sieht eine direkte Ankoppelung an den Befehlssatz einer CPU vor (Befehlssatzkopplung).
Eine zweite Variante sieht eine Ankoppelung über Tabellen im Hauptspeicher vor.
Beide sind simultan und/oder alternativ implementierbar.

Befehlssatzkoppelung

Innerhalb eines Instruktionssets (ISA) einer CPU sind für gewöhnlich freie unbenutzte Befehle vorhanden. Einer oder eine Mehrzahl dieser freien unbenutzten Befehle wird nunmehr für die Steuerung von VPUs verwendet (VPUCODE).

Durch die Dekodierung eines VPUCODEs wird eine Konfigurationseinheit (CT) einer VPU angesteuert die in Abhängigkeit des VPUCODEs bestimmte Abläufe ausführt.

Beispielsweise kann ein VPUCODE das Laden und/oder Ausführen von Konfigurationen durch die Konfigurationseinheit (CT) für eine VPU auslösen.

Kommandoübergabe an VPU

In einer erweiterten Ausführung kann ein VPUCODE über eine Übersetzungstabelle, die bevorzugt von der CPU aus aufgebaut wird, auf unterschiedliche VPU-Kommandos übersetzt werden. Die Konfigurationstabelle kann in Abhängigkeit von dem ausgeführten CPU Programm oder Codeabschnitt gesetzt werden.

Die VPU lädt nach Eintreffen eines Ladekommandos Konfigurationen aus einem eigenen oder einem z. B. mit der CPU geteilten Speicher. Insbesondere kann eine Konfiguration im Code des aktuell ausgeführten Programmes beinhaltet sein.

Nach Erhalt eines Ausführungskommandos führt eine VPU die auszuführende Konfiguration aus und die entsprechende Datenverarbeitung durch. Das Beenden der Datenverarbeitung kann durch ein Terminierungssignal (TERM) an die CPU angezeigt werden.

VPUCODE-Verarbeitung auf CPU

Bei Auftreten eines VPUCODEs können solange WarteZyklusn auf der CPU ausgeführt werden, bis das Terminierungssignal (TERM) der Beendingung der Datenverarbeitung von der VPU eintrifft.

In einer bevorzugten Ausgestaltung wird mit der Verarbeitung der nächsten Codes fortgefahren. Tritt ein weiterer VPUCODE auf, kann sodann auf die Beendigung des vorhergehenden gewartet werden, oder sämtliche gestartete VPUCODEs werden in einer Verarbeitungspipeline eingereiht, oder ein Taskwechsel wird wie nachfolgend beschrieben ausgeführt.

Die Beendigung einer Datenverarbeitung wird durch das Eintreffen des Terminierungssignal (TERM) in einem Statusregister signalisiert. Die Terminierungssignale treffen in der Reihenfolge einer möglichen Verarbeitungspipeline ein. Die Datenverarbeitung auf der CPU kann durch das Testen des Statusregisters auf das Eintreffen eines Terminierungssignales synchronisiert werden.

In einer möglichen Ausgestaltung kann, sofern eine Applikation vor dem Eintreffen von TERM z.B. durch Datenabhängigkeiten nicht fortgesetzt werden kann, ein Taskwechsel ausgelöst werden.

Coprozessor-Kopplung (lose gekoppelt)

Nach der DE 101 10 530 werden bevorzugt lose Kopplungen zwischen Prozessoren und VPUs aufgebaut, bei welchen VPUs weitestgehend als unabhängige Coprozessoren arbeiten.

Eine derartige Kopplung sieht typisch eine oder mehrere gemeinsame Datenquellen und -senken, zumeist über gemeinsame Bussysteme und/oder gemeinsame Speicher vor. Über DMAs und/oder andere Speicherzugriffskontroller werden Daten zwischen einer CPU und einer VPU ausgetauscht. Die Synchronisation der Datenverarbeitung erfolgt bevorzugt über eine Interruptsteuerung oder einen Statusabfragemechanismus (z.B. Polling).

Rechenwerk-Kopplung (eng gekoppelt)

Eine enge Ankopplung entspricht der vorab beschriebenen direkten Ankopplung einer VPU in den Befehlssatz einer CPU.

Bei einer direkten Rechenwerk-Ankopplung ist besonders auf eine hohe Rekonfigurationsperformance zu achten. Bevorzugt kann daher die Wave-Rekonfiguration nach DE 198 07 872, DE 199 26 538, DE 100 28 397 zum Einsatz kommen. Des weiteren werden die Konfigurationsworte bevorzugt nach DE 196 54 846., DE 199 26 538, DE 100 28 397, DE 102 12 621 vorab derart vorgeladen, dass bei Ausführung des Befehls die Konfiguration besonders schnell (beispielsweise mittels Wave-Rekonfigu-ration im Optimalfall innerhalb eines Taktes) konfiguriert werden kann.

Für die Wave-Reconfiguration werden bevorzugt die voraussichtlich auszuführenden Konfigurationen vorab durch den Compiler zur Compilezeit erkannt, d. h. abgeschätzt und/oder vorhergesagt, und zur Laufzeit entsprechend vorgeladen, soweit möglich. Mögliche Verfahren sind beispielsweise aus DE 196 54 846, DE 197 04 728, DE 198 07 872, DE 199 26 538, DE 100 28 397, DE 102 12 621 bekannt.

Zum Zeitpunkt der Befehlsausführung wird die oder eine entsprechende Konfiguration selektiert und ausgeführt. Auch derartige Verfahren sind nach den o.g. Schriften bekannt. Besonders bevorzugt werden Konfigurationen in Schattenkonfigurationsregister vorgeladen, wie beispielsweise aus DE 197 04 728 (Fig. 6) und DE 102 12 621 (Fig. 14) bekannt, um dann bei Abruf besonders schnell zur Verfügung zu stehen.

Datentransfers

Eine mögliche Implementierung wie beispielsweise in Figur 1 dargestellt kann unterschiedliche Datentransfers zwischen einer CPU (0101) und VPU (0102) vorsehen. Die auf der VPU auszuführenden Konfigurationen werden durch den Instruktionsdekoder (0105) der CPU selektiert, der bestimmte, für die VPU bestimmte Instruktionen erkennt und die CT (0106) derart ansteuert, dass diese die entsprechenden Konfigurationen aus einem der CT zugeordneten Speicher (0107), der insbesondere mit der CPU geshared werden oder derselbe wie der Arbeitsspeicher der CPU sein kann, in das Array aus PAEs (PA, 0108) lädt.

Es soll ausdrücklich angemerkt werden, dass in Figur 1 aus Gründen der Übersichtlichkeit nur die relevanten Komponenten (i.b. der CPU) aufgezeigt sind und eine wesentliche Zahl weiterer Komponenten und Netzwerke vorhanden sind.

Drei besonders bevorzugte einzeln oder kombiniert einsetzbare Methoden werden nachfolgend beschrieben.

Register

Bei einer Registerkopplung kann die VPU Daten aus einem CPU-Register (0103) entnehmen, verarbeiten und in ein oder das CPU-Register zurückschreiben.
Bevorzugt werden Synchronisationsmechanismen zwischen der CPU und der VPU eingesetzt.
Beispielsweise kann die VPU durch das Einschreiben der Daten in ein CPU-Register durch die CPU ein RDY-Signal (DE 196 51 075, DE 110 10 530) erhalten und daraufhin die eingeschriebenen Daten verarbeiten. Das Auslesen von Daten aus einem CPU-Register durch die CPU kann ein ACK-Signal (DE 196 51 075, DE 110 10 530) generieren, wodurch die Datenabnahme durch die CPU der VPU signalisiert wird. CPUs stellen typischerweise keine entsprechenden Mechanismen zur Verfügung.

Zwei mögliche Lösungen werden näher beschrieben:
Ein einfach zu realsierenden Ansatz ist, die Datensynchronisation über ein Statusregister (0104) durchzuführen. Beispielsweise kann die VPU das erfolgte Auslesen von Daten aus einem Register und das damit verbundene ACK-Signal (DE 196 51 075, DE 110 10 530) und/oder das Einschreiben von Daten in ein Register und das damit verbundene RDY-Signal (DE 196 51 075, DE 110 10 530) in dem Statusregister anzeigen. Die CPU testet zunächst das Statusregister und führt beispielsweise so lange Warteschleifen oder Taskwechsel aus, bis - je nach Operation - das RDY oder ACK eintraf. Danach führt die CPU den jeweiligen Registerdatentransfer aus.

In einer erweiterten Ausgestaltung wird der Befehlssatz der CPU um load/store-Instruktionen mit integrierter Statusabfrage (load_rdy, store_ack) erweitert. Beispielsweise wird bei einem store_ack nur dann ein neues Datenwort in ein CPU-Register geschrieben, wenn das Register vorher von der VPU ausgelesen wurde und ein ACK eintraf. Entsprechend liest load_rdy nur Daten aus einem CPU-Register, wenn die VPU vorher neue Daten eingeschrieben und ein RDY generiert hat.

Daten, die zu einer auszuführenden Konfiguration gehören, können successive, quasi durch Block-Moves nach dem Stand der Technik in die CPU-Register geschrieben oder aus diesen gelesen werden. Ggf. implementierte Block-Move-Instruktionen können bevorzugt durch die beschriebene integrierte RDY/ACK Statusabfrage erweitert werden.

Eine zusätzliche oder alternative Variante sieht vor, dass die Datenverarbeitung innerhalb der an die CPU gekoppelten VPU exakt gleichviele Takte benötigt wie die Datenverarbeitung innerhalb der

Rechenpipeline der CPU. Insbesondere bei modernen Hochleistungs-CPUs mit einer Vielzahl von Pipelinestufen (>20) kann dieses Konzept ideal eingesetzt werden. Der besondere Vorteil ist, dass keine besonderen Synchronisationsmechanismen wie z. B. RDY/ACK notwendig sind. Der Compiler braucht bei diesem Verfahren lediglich sicherzustellen, dass die VPU die erforderliche Anzahl an Takten einhält und ggf. die Datenverarbeitung z. B. durch das Einfügen von Verzögerungsstufen wie z. B. Registern und/oder den aus DE 110 10 530, Fig. 9/10, bekannten Fall-Through FIFOs auszubalancieren.

Eine weitere Variante ermöglicht ein unterschiedliches Laufzeitverhalten zwischen dem Datenpfad der CPU und der VPU. Dazu werden vom Compiler bevorzugt zunächst die Datenzugriffe derart umsortiert, dass eine wenigstens im wesentlichen maximale Unabhängigkeit zwischen den Zugriffen durch den Datenpfad der CPU und der VPU vorliegt. Die maximale Distanz definiert damit den maximalen Laufzeitunterschied zwischen dem CPU Datenpfad und der VPU. Mit anderen Worten wird bevorzugt durch eine Reordering Methode, wie sie per se nach dem Stand der Technik bekannt ist, der Laufzeitunterschied zwischen CPU-Datenpfad und VPU-Datenpfad ausgeglichen. Wenn der Laufzeitunterschied zu groß ist, um durch ein Umsortieren der Datenzugriffe gelöst zu werden, können per Compiler NOP-Zyklusn (also Zyklusn, in denen der CPU-Datenpfad keine Daten verarbeitet) eingefügt und/oder per Hardware so lange WarteZyklusn im CPU-Datenpfad generiert werden, bis die notwendigen Daten von der VPU in das Register geschrieben wurden. Dazu können die Register mit einem zusätzlichen Bit versehen werden, das das Vorhandensein gültiger Daten anzeigt.

Es ist ersichtlich, dass eine Vielzahl von einfachen Modifikationen und unterschiedlichen Ausgestaltungen dieser Grundverfahren möglich ist.

Die bereits erwähnte Wave-Rekonfiguration, inbesondere auch das Vorladen von Konfigurationen in Schattenkonfigurationsregister, erlaubt das sukzessive Starten eines neuen VPU-Befehls und der entsprechenden Konfiguration, sobald die Operanden des vorhergehenden VPU-Befehls aus den CPU-Registern abgenommen wurden. Die Operanden für den neuen Befehl können direkt nach Befehlsstart in die CPU-Register geschrieben werden. Entsprechend des Wave-Rekonfiguration-Verfahrens wird die VPU sukzessive mit Fertigstellung der Datenverarbeitung des vorherigen VPU-Befehls für den neuen VPU-Befehl umkonfiguriert und die neuen Operanden verarbeitet.

Buszugriffe

Weiterhin können Daten zwischen einer VPU und einer CPU durch geeignete Buszugriffe auf gemeinsame Ressourcen ausgetauscht werden.

Cache

Sofern Daten ausgetauscht werden sollen, die kurz zuvor von der CPU verarbeitet wurden und daher voraussichtlich noch im Cache (0109) der CPU liegen bzw. sofort anschliessend von der CPU verar-

beitet werden und daher sinnvollerweise in den Cache der CPU gelegt werden, werden diese bevorzugt von der VPU aus dem Cache der CPU gelesen, bzw. in den Cache der CPU geschrieben. Dies kann durch geeignete Analysen weitestgehend vorab zur Compilezeit der Applikation durch den Compiler festgestellt und es kann der Binärcode entsprechend generiert werden.

Bus
Sofern Daten ausgetauscht werden sollen, die sich voraussichtlich nicht im Cache der CPU befinden bzw. voraussichtlich nicht nachfolgend im Cache der CPU benötigt werden, werden diese bevorzugt von der VPU direkt vom externen Bus (0110) und der damit verbundenen Datenquelle (z.B. Speicher, Peripherie) gelesen, bzw. an den externen Bus und der damit verbundenen Datensenke (z.B. Speicher, Peripherie) geschrieben. Dieser Bus kann insbesondere derselbe wie der externe Bus der CPU sein (0112 & gestrichelt). Dies kann durch geeignete Analysen weitestgehend vorab zur Compilezeit der Applikation durch den Compiler festgestellt und der Binärcode entsprechend generiert werden.

Bei einem Transfer über den Bus am Cache vorbei wird bevorzugt ein Protokoll (0111) zwischen Cache und Bus implementiert, das für einen korrekten Inhalt des Caches sorgt. Beispielsweise kann das per se bekannte MESI-Protokoll nach dem Stand der Technik hierzu verwendet werden.

Cache/RAM-PAE Kopplung
Ein besonders bevorzugtes Verfahren ist die enge Kopplung von RAM-PAEs an den Cache der CPU. Dadurch können Daten schnell und effizient zwischen dem Speicher- und/oder IO- Datenbus und der VPU übertragen werden. Die Datenübertragung nach extern wird weitestgehend automatisch durch den Cachekontroller durchgeführt.

Insbesondere bei Taskwechselvorgängen, für Realtime Anwendungen und Multithreading CPUs bei dem Wechsel von Threads erlaubt dieses Verfahren einen schnellen und unkomplizierten Datenaustausch.

Zwei grundlegende Verfahren stehen zur Verfügung:

a) RAM-PAE / Cache Kopplung
Die RAM-PAE überträgt Daten z. B. zum Lesen und/oder Schreiben von externen und insbesondere Hauptspeicher-Daten direkt zu und/oder vom Cache. Dazu kann bevorzugt ein separater Datenbus nach DE 196 54 595 und DE 199 26 538 verwendet werden, über welchen unabhängig von der Datenverarbeitung innerhalb der VPU und insbesondere auch automatisch gesteuert, z.B. durch eigenständige Adressgeneratoren, Daten zu oder von dem Cache übertragen werden können.

b) RAM-PAE als Cache-Slice
In einer besonders bevorzugten Ausgestaltung weisen die RAM-PAEs keinen internen Speicher auf, sondern sind direkt an Blöcke (Slices) des Caches gekoppelt. Mit anderen Worten, beinhalten die RAM-

PAEs lediglich die Busansteuerungen für die lokalen Busse, sowie eventuelle Zustandsmaschinen und/oder eventuelle Adressgeneratoren, der Speicher befindet sich jedoch innerhalb einer Cachespeicherbank auf die die RAM-PAE direkten Zugriff hat. Jede RAM-PAE besitzt eine eigene Slice innerhalb des Caches und kann unabhängig und insbesondere gleichzeitig zu den anderen RAM-PAEs und/oder der CPU auf den Cache, bzw. den eigenen Slice zugreifen. Dies kann einfach dadurch realisiert werden, dass der Cache aus mehreren unabhängigen Banks (Slices) aufgebaut ist.

Sofern der Inhalt einer Cache-Slice durch die VPU verändert wurde, kann dieser bevorzugt als "dirty" markiert werden, woraufhin der Cachekontroller diesen automatisch in den externen und/oder Haupt-Speicher zurückschreibt.

Für manche Anwendungen kann zusätzlich eine Write-Through Strategie implementiert oder gewählt sein. Hierbei werden neu von der VPU in die RAM-PAEs geschrieben Daten direkt mit jedem Schreibvorgang in den externen und/oder Haupt-Speicher zurückgeschrieben. Dadurch entfällt zusätzlich die Notwendigkeit Daten mit "dirty" zu markieren und diese bei einem Task- und/oder Threadwechsel in den externen und/oder Haupt-Speicher zurückzuschreiben.

In beiden Fällen kann es sinnvoll sein, bestimmte Cache-Bereiche für die RAM-PAE/Cache Kopplung für den Zugriff durch die CPU zu sperren.

An die beschriebene Architektur, insbesondere direkt an die VPU, kann ein FPGA (0113) gekoppelt sein, um feingranulare Datenverarbeitung zu ermöglichen und/oder ein flexible adaptierbare Interface (0114) (z. B. diverse serielle Schnittstellen (V24, USB, etc.), diverse parallele Schnittstellen, Festplattenschnittstellen, Ethernet, Telekommunikationsschnittstellen (a/b, T0, ISDN, DSL, etc.) zu weiteren Baugruppen und/oder dem externen Bussystem (0112) zu ermöglichen. Der FPGA kann dabei aus der VPU-Architektur, insbesondere durch die CT, und/oder durch die CPU konfiguriert werden. Der FPGA kann statisch, also ohne Rekonfiguration zur Laufzeit und/oder dynamisch, also mit Rekonfiguration zur Laufzeit, betrieben werden.

FPGAs in ALUs

In einer "prozessororientierteren" Ausgestaltung können innerhalb einer ALU-PAE FPGA-Elemente aufgenommen werden. Dabei kann ein FPGA-Datenpfad parallel zu der ALU gekoppelt sein, oder in einer bevorzugten Ausgestaltung der ALU vor- oder nachgeschaltet sein.

Innerhalb von in Hochsprachen wie C geschriebenen Algorithmen treten bit-orientierte Operationen zumeist sehr sporadisch auf und sind nicht besonders aufwendig. Daher ist ein FPGA-Aufbau von einigen wenigen Zeilen von Logikelementen jeweils miteinander verkoppelt durch eine Zeile von Verdrahtungskanälen hinreichend. Eine derartige Struktur lässt sich kostengünstig und einfach program-

mierbar in die ALU einbinden. Ein wesentlicher Vorteil für die nachfolgend erläuterten Programmiermethoden kann sein, dass die Durchlaufzeit durch die FPGA-Struktur derart begrenzt ist, dass sich das Laufzeitverhalten der ALU nicht verändert. Register brauchen nur zur Speicherung von Daten für deren Einbeziehung als Operanden in den im nächsten Takt nachfolgenden Verarbeitungszyklus zulässig zu sein.

Besonders von Vorteil ist die Implementierung von optional hinzukonfigurierbaren Register, um ein sequentielles Verhalten der Funktion durch beispielsweise Pipelining herzustellen. Dies ist besonders dann von Vorteil, wenn Rueckkopplungen in dem Code fuer die FPGA-Struktur vorkommen. Diese kann der Compiler dann ueber das Einschalten solcher Register per Konfiguration abbilden und somit sequentiellen Code korrekt abbilden. Der Zustandsmaschine der PAE, die deren Abarbeitung steuert, wird die Anzahl der eingefuegten Register per Konfiguration mitgeteilt, sodass diese ihre Steuerung, insbesondere auch der PAE-externen Datenuebrtragung, auf die erhoehte Latenzzeit abstimmen kann.

Von besonderem Vorteil ist ein Aufbau der FPGA-Struktur, die ohne Konfiguration, also z. B. nach einem Reset, automatisch auf neutral geschaltet ist, d. h. die Eingangsdaten ohne jegliche Modifikation durchleitet. Damit werden, bei nicht verwendeten FPGA-Strukturen keinerlei Konfigurationdaten zu deren Einstellung benötigt und somit Konfigurationszeit und -datenplatz in den Konfigurationsspeichern eingespart.

Betriebssystemmechanismen

Die beschriebenen Verfahren sehen zunächst keinen besonderen Mechanismus für die Unterstützung von Betriebssystemen vor. Es ist nämlich bevorzugt sicherzustellen, dass ein auszuführendes Betriebssystem sich entsprechend des Status einer zu unterstützenden VPU verhält. Insbesondere sind Scheduler erforderlich.

Bei einer engen Rechenwerkkopplung wird bevorzugt das Statusregister der CPU abgefragt, in welches die angekoppelte VPU ihren Datenverarbeitungsstatus (Terminierungssignal) einträgt. Soll eine weitere Datenverarbeitung an die VPU übertragen werden, und die VPU hat die vorherige Datenverarbeitung noch nicht beendet, wird gewartet oder bevorzugt ein Taskwechsel ausgeführt.

Grundsätzlich kann die Ablaufsteuerung einer VPU direkt von einem auf der CPU ausgeführten Programm erfolgen, das quasi das Hauptprogramm darstellt, das bestimmte Unterprogramme auf die VPU auslagert.

Für eine Coprozessorkopplung werden bevorzugt über das Betriebssystem, i.b. den Scheduler gesteuerte Mechanismen verwendet, wobei grundsätzlich die Ablaufsteuerung einer VPU direkt von einem auf

der CPU ausgeführten Programm erfolgen kann, das quasi das Hauptprogramm darstellt, das bestimmte Unterprogramme auf die VPU auslagert:
Ein einfacher Scheduler kann nach Übertragung einer Funktion auf eine VPU

1. das aktuelle Hauptprogramm auf der CPU weiterlaufen lassen, sofern dieser unabhängig und parallel zur Datenverarbeitung auf einer VPU ablaufen kann;
2. sofern oder sobald das Hauptprogramm auf die Beendigung der Datenverarbeitung auf der VPU warten muss, schaltet der Taskscheduler auf einen anderen Task (z.B. ein anderes Hauptprogramm) um. Die VPU kann dabei unabhängig des gerade aktuellen CPU-Tasks im Hintergrund weiterarbeiten.

Jeder neu aktivierte Task muss, sofern er die VPU verwendet, vor Verwendung prüfen, ob diese für eine Datenverarbeitung zur Verfügung steht oder aktuell noch Daten verarbeitet; dann muss entweder auf die Beendigung der Datenverarbeitung gewartet oder bevorzugt der Task gewechselt werden.

Ein einfaches und dennoch leistungsfähiges Verfahren kann durch sogenannte Descriptor Tables aufgebaut werden, die bespielsweise folgendermassen realisiert werden können:
Jeder Task generiert zum Aufruf der VPU eine oder mehrere Tabelle(n) (VPUPROC) mit einem geeigneten festgelegten Datenformat in dem ihm zugewiesenen Speicherbereich. Diese Tabelle beeinhaltet sämtliche Steuerinformation für eine VPU, wie z. B. das auszuführende Programm / die auszuführende(n) Konfiguration(en) (oder Zeiger auf die entsprechenden Speicherstellen) und/oder Speicherstelle(n) (oder jeweils Zeiger darauf) und/oder Datenquellen (oder jeweils Zeiger darauf) der Eingangsdaten und/oder die Speicherstelle(n) (oder jeweils Zeiger darauf) der Operanden oder der Ergebnisdaten.

Entsprechend Figur 2 kann sich im Speicherbereich des Betriebssystems beispielsweise eine Tabelle oder verkettete Liste (LINKLIST, 0201) befinden, die auf sämtliche VPUPROC-Tabellen (0202) in der Reihenfolge ihrer Erstellung und/oder ihres Aufrufs zeigt.

Die Datenverarbeitung auf der VPU läuft nunmehr derart ab, dass ein Hauptprogramm einen VPUPROC erstellt und über das Betriebssystem die VPU aufruft. Das Betriebssystem erstellt einen Eintrag in der LINKLIST. Die VPU arbeitet die LINKLIST ab und führt die jeweils referenzierten VPUPROC aus. Die Beendigung einer jeweiligen Datenabarbeitung wird jeweils durch einen entsprechenden Eintrag in die LINKLIST und/oder VPUCALL Tabelle angezeigt. Alternativ können Interrupts von der VPU zur CPU als Anzeige und ggf. auch zum Austausch des VPU-Status verwendet werden.

Die VPU arbeitet in diesem erfindungsgemäß bevorzugten Verfahren weitgehend unabhängig von der CPU. Insbesondere kann die CPU und die VPU unabhängige und unterschiedliche Tasks je Zeiteinheit aus-

führen. Das Betriebssystem und/oder die jeweiligen Task müssen lediglich die Tabellen (LINKLIST bzw. VPUPROC) überwachen.

Alternativ kann auch auf die LINKLIST verzichtet werden, indem die VPUPROCs untereinander durch Pointer verkettet werden, wie es z.B. aus Listen bekannt ist. Abgearbeitete VPUPROCs werden aus der Liste entfernt, neue in die Liste eingefügt. Das Verfahren ist Programmierern bekannt und muss daher nicht weitergehend ausgeführt werden.

Multithreading/Hyperthreading

Von besonderem Vorteil ist die Verwendung von Multithreading und/oder Hyperthreading Technologien, bei welchen ein - bevorzugt in Hardware implementierter - Scheduler feingranular Applikationen und/oder Applikationsteile (Thread) auf Ressourcen innerhalb des Prozessors verteilt. Dabei wird der VPU-Datenpfad als eine Ressource fuer den Scheduler betrachtet. Eine saubere Trennung des CPU-Datenpfades und des VPU-Datenpfades ist durch Implementierung von Multithreading und/oder Hyperthreading Technologien im Compiler bereits per Definition gegeben. Zusaetzlich entsteht der Vorteil, dass bei belegter VPU-Ressource innerhalb eines Task einfach zu einem anderen Task gewechselt werden kann und somit eine bessere Auslastung der Ressourcen erfolgt. Gleichzeitig wird die parallele Ausnutzung von CPU-Datenpfad und VPU-Datenpfad zugleich beguenstigt.
Insoweit stellt Multithreading und/oder Hyperthreading ein gegenueber den vorstehend beschriebenen LINKLIST zu bevorzugendes Verfahren dar.

Besonders performanceeffizient arbeiten die beiden Verfahren, wenn als VPU eine Architektur zum Einsatz kommt, die eine mit der Datenverarbeitung überlagerte Rekonfiguration zulässt, wie z. B. die Wave-Reconfiguration nach DE 198 07 872, DE 199 26 538, DE 100 28 397.
Damit ist es möglich, eine neue Datenverarbeitung und eine ggf. damit verbundene Rekonfiguration sofort nach lesen der letzten Operanden aus den Datenquellen zu starten. Mit anderen Worten ist für die Synchronisation nicht mehr das Beenden der Datenverarbeitung, sondern das Lesen der letzten Operanden erforderlich. Dadurch wird die Performance der Datenverarbeitung erheblich gesteigert.

Figur 3 zeigt einen möglichen internen Aufbau eines Mikroprozessors oder Mikrokontrollers. Dargestellt ist der Kern (0301) eines Mikrokontrollers oder Mikroprozessors. Der beispielhafte Aufbau beinhaltet weiterhin eine Load/Store Einheit zum Übertragen der Daten zwischen dem Kern und dem externen Speicher und/oder den Peripheriegeräten. Die Übertragung findet über das Interface 0303 statt, an das weitere Einheiten, wie MMUs, Caches, etc. angekoppelt sein koennen.
In einer Prozessorarchitektur nach dem Stand der Technik überträgt die Load/Store Einheit die Daten zu oder von einem Registersatz

(0304), welcher sodann die Daten für die interne Weiterverarbeitung zwischenspeichert. Die interne Weiterverarbeitung findet in einem oder mehreren Datenpfaden statt, die jeweils identisch oder auch unterschiedlich ausgestaltet sein können (0305). Insbesondere können auch mehrere Registersätze vorhanden sein, wobei diese wiederum ggf. an unterschiedliche Datenpfade angekoppelt sein können (z.B. Integer-Datenpfade, Floating-Point-Datenpfade, DSP-Datenpfade / Multiply-Accumulate-Einheiten).

Datenpfade entnehmen typischerweise Operanden aus der Registereinheit und schreiben die Ergebnisse nach der Datenverarbeitung wieder an die Registereinheit zurück. Dem Kern zugeordnet (oder im Kern beinhaltet) ist eine Instruktionsladeeinheit (Opcode-Fetcher, 0306), die die Programmcodebefehle aus dem Programmspeicher lädt, übersetzt und sodann die notwendigen Arbeitsschritte innerhalb des Kerns ansteuert. Das Holen der Befehle geschieht über ein Interface (0307) zu einem Codespeicher, dem ggf. MMUs, Caches, etc. zwischengeschaltet sein können.

Dem Datenpfad 0305 parallelgeschaltet ist der VPU-Datenpfad (0308), der auf den Registersatz 0304 lesend und über die nachfolgend beschriebene Daten-Register-Zuordungseinheit (0309) schreibend zugreifen kann. Der Aufbau eines VPU Datenpfades ist beispielsweise aus DE 196 51 075, DE 100 50 442, DE 102 06 653 und mehreren Veröffentlichungen der Anmelderin bekannt.

Der VPU-Datenpfad wird über den Konfigurationmanager (CT) 0310 konfiguriert, der die Konfigurationen über einen Bus 0311 aus einem externen Speicher lädt. Der Bus 0311 kann identisch mit 0307 sein, wobei je nach Ausgestaltung zwischen 0311 und 0307 und/oder den Speicher ein oder mehrere Caches geschaltet sein können.

Welche Konfiguration zu einem bestimmten Zeitpunkt konfiguriert und ausgeführt werden soll, wird über besondere OpCodes durch den OpCode-Fetcher 0306 definiert. Dazu kann einer Reihe von für den VPU-Datenpfad reservierten OpCodes eine Anzahl von möglichen Konfigurationen zugeordnet werden. Die Zuordnung kann über eine umprogrammierbare Lookup-Tabelle (siehe 0106), die 0310 vorgeschaltet ist, erfolgen, so dass die Zuordnung frei programmierbar und innerhalb der Applikation veränderbar ist.

In einer applikationsabhängig möglichen Ausgestaltung kann bei einem Aufruf einer VPU-Datenpfadkonfiguration das Zielregister der Datenberechnung in der Daten-Register-Zuordungs-einheit (0309) verwaltet werden. Dazu wird das vom OpCode definierte Zielregister in einen Speicher bzw. Register (0314) geladen, der - um mehrere VPU-Datenpfadaufrufe direkt nacheinander und ohne Berücksichtigung der Abarbeitungszeit der jeweiligen Konfiguration zuzulassen - als FIFO ausgestaltet sein kann. Sobald eine Konfiguration die Ergebnisdaten liefert, werden diese mit der jeweils zugeordneten Registeradresse verknüpft (0315) und das entsprechende Register in 0304 selektiert und beschrieben.

Damit können eine Vielzahl von VPU-Datenpfadaufrufe direkt hintereinander und insbesondere überlagernd erfolgen. Es ist lediglich sicherzustellen, beispielsweise durch Compiler oder Hardware, dass die Operanden und Ergebnisdaten derart gegenüber der Datenverarbeitung in dem Datenpfad 0305 umsortiert sind, dass keine Störung durch unterschiedliche Laufzeiten in 0305 und 0308 auftreten.

Wenn der Speicher bzw. FIFO 0314 voll ist, wird die Verarbeitung einer ggf. neuen Konfiguration für 0308 verzögert. Sinnvollerweise kann 0314 so viele Registerdaten aufnehmen, wie 0308 Konfigurationen in einem Stack (siehe DE 197 04 728, DE 100 28 397, DE 102 12 621) vorladen kann. Über den Speicher 0314 können zusätzlich zu einer Verwaltung durch den Compiler auch die Datenzugriffe auf den Registersatz 0304 gesteuert werden.

Findet ein Zugriff auf ein Register, das in 0314 vermerkt ist, statt, kann dieser so lange verzögert werden, bis das Register beschrieben wurde und dessen Adresse aus 0314 entfernt wurde.

Alternativ und bevorzugt können die einfachen Synchronisationsmethoden nach 0103 verwendet werden, wobei ein besonderes synchrones Datenempfangsregister in dem Registersatz 0304 vorgesehen sein kann, auf das nur dann lesend zugegriffen werden kann, wenn der VPU-Datenpfad 0308 zuvor neue Daten in das Register geschrieben hat; umgekehrt können Daten von dem VPU-Datenpfad nur dann geschrieben werden, wenn die vorherigen Daten gelesen wurden. Insoweit kann 0309 ersatzlos entfallen.

Wird eine VPU-Datenpfadkonfiguration aufgerufen die bereits konfiguriert ist, findet keine Neukonfiguration mehr statt. Daten werden sofort zur Verarbeitung aus dem Registersatz 0304 in den VPU-Datenpfad übertragen und verarbeitet. Der Konfigurationmanager speichert die aktuell geladene Konfigurationskennnummer in einem Register und vergleicht diese mit der von zu ladenden Konfigurationskennnummer, die beispielsweise ueber eine Lookup-Tabelle (siehe 0106) an 0310 übertragen wird. Nur wenn die Nummern nicht übereinstimmen, wird die aufgerufene Konfiguration neu konfiguriert.

Die Load/Store-Einheit ist in Fig. 3 nur schematisch und grundlegend dargestellt, eine bevorzugte Ausführung wird in Fig. 4 und 5 ausführlich dargestellt. Über ein Bussystem 0312 kann der VPU-Datenpfad (0308) direkt Daten mit der Load/Store Einheit und/oder dem Cache übertragen, über einen weiteren applikationsabhängig möglichen Datenpfad 0313 können Daten direkt zwischen VPU-Datenpfad (0308) und Peripheriegeräten und/oder externem Speicher übertragen werden.

Figur 4 zeigt eine besonders bevorzugte Ausgestaltung der Load/Store-Einheit.
Ein wesentliches Datenverarbeitungsprinzip der VPU Architektur sieht an das Array aus ALU-PAEs gekoppelte Speicherbloecke vor, die quasi als Registersatz für Datenblöcke dienen. Das Verfahren ist aus DE 196 54 846, DE 101 39 170, DE 199 26 538, DE 102 06 653

bekannt. Hierzu bietet es sich, wie nachfolgend beschrieben, an, LOAD und STORE Befehle als Konfiguration innerhalb der VPU abzuarbeiten, was eine Verschaltung der VPU mit der Load/Store Einheit (0401) der CPU überflüssig macht. Mit anderen Worten generiert die VPU ihre Lese- und Schreibzugriffe selbst, wodurch ein direkter Anschluss (0404) an den externen und/oder Haupt-Speicher sinnvoll ist. Dieser geschieht bevorzugt über einen Cache (0402), der derselbe wie der Datencache des Prozessors sein kann. Die Load/Store-Einheit des Prozessors (0401) greift direkt und parallel zu der VPU (0403) auf den Cache zu ohne - im Gegensatz zu 0302 - einen Datenpfad für die VPU aufzuweisen.

Figur 5 zeigt besonders bevorzugte Ankopplungen der VPU an den externen und/oder Haupt- Speicher über einen Cache.
Die einfachste Verbindungsmethode ist über einen IO-Anschluss der VPU, wie beispielsweise aus DE 196 51 075.9-53, DE 196 54 595.1-53, DE 100 50 442.6, DE 102 06 653.1 bekannt, über welchen Adressen und Daten zwischen Peripherie und/oder Speicher und der VPU übertragen werden. Besonders leistungsfähig sind aber direkte Ankopplungen zwischen den RAM-PAEs und dem Cache, wie aus DE 196 54 595 und DE 199 26 538 bekannt. Beispielhaft für ein rekonfigurierbares Datenverarbeitungselement ist eine PAE dargestellt, aufgebaut aus einer Hauptdatenverarbeitungseinheit (0501), die typischerweise als ALU, RAM, FPGA, IO-Anschluss ausgestaltet ist, und zwei seitlichen Datenübertragungseinheiten (0502, 0503), die ihrerseits eine ALU-Struktur und/oder Registerstruktur aufweisen können. Desweiteren sind die Array-internen, zur PAE gehörenden horizontalen Bussysteme 0504a und 0504b dargestellt.

In Figur 5a sind RAM-PAEs (0501a), die jeweils einen eigenen Speicher beinhalten gemäß DE 196 54 595 und DE 199 26 538 über einen Multiplexer 0511 an einen Cache 0510 gekoppelt. Cachekontroller und Verbindungsbus des Caches zum Hauptspeicher sind nicht dargestellt. Die RAM-PAEs weisen bevorzugt einen separaten Datenbus (0512) mit eigenen Adressgeneratoren (siehe auch DE 102 06 653) auf, um Daten selbständig in den Cache übertragen zu können.

Figur 5b zeigt eine optimierte Variante. 0501b sind keine vollwertigen RAM-PAEs, sondern beinhalten nur die Bussysteme und seitlichen Datenübertragungseinheiten (0502, 0503). Anstatt des integrierten Speichers in 0501 ist lediglich eine Busverbindung (0521) zu dem Cache 0520 implementiert. Der Cache ist in mehrere Segmente unterteilt 05201, 05202 ... 0520n, die jeweils einem 0501b zugeordnet sind und für diesen 0501b bevorzugt exklusiv reserviert sind. Der Cache stellt somit quasi die Menge aller RAM-PAEs der VPU und den Datencache (0522) der CPU dar.

Die VPU schreibt ihre internen (Register-) Daten direkt in den Cache, bzw. liest diese direkt aus dem Cache.
Veränderte Daten können mit "dirty" markiert werden, woraufhin der nicht eingezeichnete Cachekontroller diese automatisch im Hauptspeicher updated. Alternativ stehen Write-Through Verfahren zur Verfügung, bei denen veränderte Daten direkt in den Hauptspeicher

geschrieben werden und das verwalten des "dirties" überflüssig wird.

Die direkte Kopplung nach Figur 5b ist besonders bevorzugt, da sie äusserst flächeneffizient ist und einfach durch die VPU zu handhaben ist, da die Cachekontroller die Datenübertragung zwischen Cache - und somit der RAM-PAE - und Hauptspeicher automatisch übernehmen.

Figur 6 zeigt die Ankopplung einer FPGA-Struktur in einen Datenpfad am Beispiel der VPU Architektur.
0501 ist der Hauptdatenpfad einer PAE. Bevorzugt werden FPGA-Strukturen direkt nach den Eingangsregistern (vgl. PACT02, PACT22) eingefuegt (0611) und/oder direkt vor den Ausgang des Datenpfades auf das Bussystem eingefuegt (0612).
Eine moegliche FPGA-Struktur ist in 0610 dargestellt, der Aufbau ist an PACT13 Figur 35 angelehnt.
Die FPGA Struktur ist ueber einen Dateneingang (0605) und einen Datenausgang (0606) in die ALU eingekoppelt. Jeweils abwechselnd sind
a) in einer Zeile (0601) Logikelemente angeordnet, die bitweise logische (AND, OR, NOT, XOR, etc) Operationen ueber eingehende Daten durchfuehren. Diese Logikelemente koennen zusaetzlich lokale Busverbindungen aufweisen, ebenso koennen Register zur Datenspeicherung in den Logikelementen vorgesehen sein.
b) in einer Zeile (0602) Speicherelemente angeordnet, die bitweise Daten der Logikelemente speichern. Ihre Aufgabe ist bei Bedarf die zeitliche Entkopplung - also das zyklische Verhalten - eines sequentiellen Programmes darzustellen, sofern dies von Compiler gefordert wird. Mit anderen Worten wird durch diese Registerstufen das sequentielle Verhalten eines Programmes in Form einer Pipeline innerhalb von 0610 nachgebildet.

Jeweils zwischen den Elementen 0601 und 0602 befinden sich horizontale konfigurierbare Signalnetzwerke, die entsprechend der bekannten FPGA Netzwerke aufgebaut sind. Diese erlauben eine horizontale Verschaltung und Uebertragung von Signalen.

Es kann zusaetzlich ein vertikales Netzwerk (0604) zur Signaluebertragung vorgesehen sein, das ebenfalls entsprechend der bekannten FPGA Netzwerke aufgebaut aufgebaut ist. Mittels dieses Netzwerks lassen sich Signale an mehreren Zeilen von Elementen 0601 und 0602 vorbeiuebertragen.

Da die Elemente 0601 und 0602 typischerweise bereits eine Anzahl vertikaler Bypass-Signal-Netzwerke aufweisen ist 0604 nur optional und bei einer grossen Anzahl von Zeilen erforderlich.

Zur Abstimmung der Zustandsmaschine der PAE auf die jeweils konfigurierte Tiefe der Pipeline in 0610, also die Anzahl (NRL) der einkonfigurierten Registerstufen (0602) zwischen dem Eingang (0605) und dem Ausgang (0606), ist ein Register 0607 implementiert, in welches NRL konfiguriert wird. Anhand dieser Daten

stimmt die Zustandsmaschine die Generierung der PAE internen Steuerzyklen und insbesondere auch der Handshake-Signale (PACT02, PACT16, PACT18) fuer die PAE-externen Bussysteme ab.

Weitere moegliche FPGA-Strukturen sind beispielsweise von Xilinx und Altera bekannt, wobei diese bevorzugt eine Registerstruktur nach 0610 aufweisen.

Figur 7 zeigt mehrere Strategien Code-Kompatibilitaet zwischen unterschiedlich grossen VPUs zu erzielen:
0701 ist eine ALU-PAE- (0702) RAM-PAE- (0703) Anordnung die eine moegliche "kleine" VPU definiert. Es soll im folgenden davon ausgegangen werden, dass Code fuer diese Struktur generiert wurde und nunmehr auf anderen groesseren VPUs abgearbeitet werden soll.

Ein erster moeglicher Ansatz ist den Code fuer die neue Ziel-VPU neu zu compilieren. Dies bietet insbesondere den Vorteil, dass eventuell in einer neuen Ziel-VPU nicht mehr vorhandene Funktionen dadurch nachgebildet werden, dass der Compiler Macros fuer diese Funktionen instanttiert, die dann die urspruengliche Funktion nachbilden. Die Nachbildung kann entweder durch die Verwendung mehrerer PAEs erfolgen und/oder durch den Einsatz von Sequenzern wie nachfolgend beschrieben (z.B. fuer Division, Floating-Point, komplexe Mathematik, etc) und beispielsweise aus PACT02 bekannt. Der klare Nachteil des Verfahrens ist, dass die Binaer-Kompatibilitaet verloren geht.

Die in Figur 7 beschriebenen Verfahren erhalten die Binaerkode-Kompatibilitaet.
Ein erstes einfaches Verfahren sieht das einfuegen von "Wrapper"-Code vor (0704), der die Bussysteme zwischen einem kleinen ALU-PAE Array und den RAM-PAEs verlaengert. Der Code beinhaltet lediglich die Konfiguration fuer die Bussysteme und wird in den bestehende Binaerkode, beispielswiese zur Konfigurationszeit und/oder zur Ladezeit aus einem Speicher eingefuegt.
Der einzige Nachteil des Verfahren ist, dass eine laenger Uebertragungszeit der Informationen ueber die verlaengerten Bussysteme entsteht. Bei vergleichsweise niederen Frequenzen kann dies vernachlaessigt werden (Fig. 7a a)).
In Figur 7a b) ist eine einfache optimierte Variante dargestellt in welcher die Verlaengerung der Bussysteme ausgeglichen und damit weniger frequenzkritisch ist, das sich die Laufzeit fuer das Wrapper-Bussystem gegenueber Fig. 7a a) halbiert.

Fuer hoehere Frequenzen ist das Verfahren nach Fig. 7b einsetzbar, bei welcher eine groessere VPU ein Superset der kompatiblen kleinen VPU (0701) darstellt und die kompletten Strukturen von 0701 repliziert werden. Dadurch ist eine direkte Binaerkompatibilitaet einfach gegeben.

Ein optimales Verfahren sieht nach Figur 7c zusaetzliche Hochgeschwindigkeitsbussysteme vor, die einen Anschluss (0705) an jede PAE oder jeweils an eine Gruppe von PAEs aufweisen. Derartige Bus-

systeme sind aus anderen Patentanmeldungen der Anmelderin bekannt, beispielswiese aus PACT07. Ueber die Anschluesse 0705 werden die Daten auf ein Hochgeschwindigkeitsbussystem (0706) uebertragen, das diese dann ueber eine grosse Strecke performanceeffizient uebertraegt. Als derartige Hochgeschwindigkeitsbussysteme koennen beispielsweise Ethernet, RapidIO, USB, AMBA, RAMBUS und andere Industriestandards verwendet werden.

Der Anschluss an das Hochgeschwindigkeitsbussystem kann entweder durch einen Wrapper wie fuer Fig. 7a beschrieben eingefuegt werden, oder architektonisch bereits fuer 0701 vorgesehen sein. In diesem Fall wird bei 0701 der Anschluss einfach direkt an die benachbarte Zelle weitergeleitet und nicht verwendet. Die Hardware abstrahiert das nicht-Vorhandensein des Bussystems.

Im vorstehenden wurde allgemein auf die Kopplung zwischen einem Prozessor und einer VPU bzw., allgemeiner, einer insbesodnere zur Laufzeit ganz und/oder partiell und/oder schnell, d.h. in wenigen TaktZyklusn vollständig, rekonfigurierbaren Einheit Bezug genommen. Diese Kopplung kann durch die Verwendung bestimmter Betriebsverfahren respektive durch dem Betrieb vorhergehenden geeignete Kompilierung unterstützt und/oder erreicht werden. Geeignete Kompilierung kann dabei wie erforderlich auf nach dem Stand der Technik bestehender und/oder auf erfindungsgemäß verbesserte Hardware Rückgriff nehmen.

Parallelisierende Compiler nach dem Stand der Technik verwenden für gewöhnlich spezielle Konstrukte, wie Semaphore und/ oder andere Verfahren zur Synchronisation. Dabei werden typischerweise technologiespezifische Verfahren verwendet. Bekannte Verfahren sind aber nicht geeignet, um funktional spezifizierte Architekturen mit dem zugehörigen Zeitverhalten und imperativ spezifizierte Algorithem zu kombinieren. Daher liefern die verwendeten Methoden nur in Spezialfällen zufriedenstellende Lösungen.

Compiler für rekonfigurierbare Architekturen, insbesondere rekonfigurierbare Prozessoren, verwenden für gewöhnlich Makros, die speziell für die bestimmte rekonfigurierbare Hardware erstellt wurden, wobei für die Erstellung der Makros für zumeist Hardwarebeschreibungssprachen (wie z. B. Verilog, VHDL, System-C) verwendet werden. Diese Makros werden dann von einer gewöhnlichen Hochsprache (z. B. C, C++) aus dem Programmfluss heraus aufgerufen (instantiiert).

Compiler für Parallelrechner sind bekannt, die auf einer grobgranularen Struktur, zumeist basierend auf kompletten Funktionen oder Threads, Programmteile auf mehrere Prozessoren abbilden. Weiterhin sind vektorisierende Compiler bekannt, die eine weitgehende lineare Datenverarbeitung, wie z. B. Berechnungen großer Ausdrücke in eine vektorisierte Form umwandeln und damit die Berechnung auf su-

perskalaren Prozessoren und Vektorprozessoren (z. B. Pentium, Cray) ermöglichen.

Dieses Patent beschreibt daher weiter ein Verfahren zur automatischen Abbildung von funktional oder imperativ formulierten Rechenvorschriften auf unterschiedliche Zieltechnologien, insbesondere auf ASICs, rekonfigurierbare Bausteine (FPGAs, DPGAs, VPUs, ChessArray, KressArray, Chameleon, etc.; im folgenden unter dem Begriff VPU zusammengefaßt), sequentielle Prozessoren (CISC-/RISC-CPUs, DSPs, etc.; im folgenden unter dem Begriff CPU zusammengefaßt) und parallele Rechnersysteme (SMP, MMP, etc.).

VPUs bestehen grundsätzlich aus einer mehrdimensionalen homogenen oder inhomogenen, flachen oder hierarchischen Anordnung (PA) von Zellen (PAEs), die beliebige Funktionen, i. b. logische und/oder arithmetische Funktionen (ALU-PAEs) und/oder Speicherfunktionen (RAM-PAEs) und/oder Netzwerkfunktionen ausführen können. Den PAEs ist eine Ladeeinheit (CT) zugeordnet, die die Funktion der PAEs durch Konfiguration und gegebenenfalls Rekonfiguration bestimmt.

Das Verfahren basiert auf einem abstrakten parallelen Maschinenmodell, das neben dem endlichen Automaten auch imperative Problemspezifikationen integriert und eine effiziente algorithmische Ableitung einer Implementierung auf unterschiedliche Technologien ermöglicht.

Die Erfindung ist eine Weiterentwicklung der Compilertechnologie nach DE 101 39 170.6, welche besonders die enge XPP-Anbindung an einen Prozessor innerhalb dessen Datenpfaden beschreibt und einen dafür besonders geeigneten Compiler offenbart, der zudem auch XPP Standalonesysteme ohne enge Prozessorkopplung verwendet werden.

Zumindest folgende Compilerklassen sind nach dem Stand der Technik bekannt: Klassische Compiler, die häufig Stack-Maschinen-Code generieren und für sehr einfache Prozessoren geeignet waren, die im Wesentlichen als normale Sequenzer ausgestaltet sind. (vgl. N.Wirth, Compilerbau, Teubner Verlag).

Vektorisierende Compiler bauen weitgehend linearen Code, der auf spezielle Vektorrechner oder stark gepipelinede Prozessoren abgestimmt ist. Ursprünglich waren diese Compiler für Vektorrechner wie CRAY verfügbar. Moderne Prozessoren wie Pentium benötigen aufgrund der langen Pipelinestruktur ähnliche Verfahren. Da die einzelnen Rechenschritte vektorisiert (gepipelined) ablaufen, ist der Code sehr viel effizienter. Allerdings bereitet der bedingte Sprung Probleme für die Pipeline. Daher ist eine Sprungvorhersage sinnvoll, die ein Sprungziel annimmt. Ist die Annahme falsch, muss jedoch die gesamte Verarbeitungspipeline gelöscht werden. Mit anderen Worten ist jeder Sprung für diese Compiler problematisch, eine Parallelverarbeitung im eigentlichen Sinn ist nicht gegeben. Sprungvorhersagen und ähnliche Mechanismen erfordern einen erheblichen Zusatzaufwand an Hardware.

Grobgranulare parallele Compiler existieren im eigentlichen Sinne kaum, die Parallelität wird typischerweise durch den Programmierer oder das Betriebssystem markiert und verwaltet, also beipielsweise bei MMP-Computersystemen wie verschiedene IBM Architekturen, ASCII Red etc. zumeist auf Thread-Ebene durchgeführt. Ein Thread ist ein weitgehend unabhängiger Programmblock oder gar ein anderes Programm. Threads sind daher grobgranular einfach zu parallelisieren. Die Synchronisation und Datenkonsistenz ist vom Programmierer bzw. dem Betriebssystem sicherzustellen. Dies ist aufwendig zu programmieren und erfordert einen wesentlichen Anteil der Rechenleistung eines Parallelrechners. Zudem ist durch diese grobe Parallelisierung nur ein Bruchteil der eigentlich möglichen Parallelität tatsächtlich nutzbar.

Feingranulare parallele (z. B. VLIW) Compiler versuchen, die Parallelität feingranular in VLIW-Rechenwerke abzubilden, die mehrere Rechenoperationen in einem Takt parallel ausführen können, aber einen gemeinsamen Registersatz besitzen. Ein wesentliches Problem stellt dieser limitierte Registersatz dar, da er die Daten für sämtliche Rechenoperationen bereitstellen muss. Zudem erschweren Datenabhängigkeiten und inkonsistente Lese/Schreiboperationen (LOAD/STORE) die Parallelisierung.

Rekonfigurierbare Prozessoren weisen eine große Anzahl an unabhängigen Rechenwerken auf. Diese sind nicht durch einen gemeinsamen Registersatz, sondern durch Busse miteinander verbunden. Dadurch lassen sich einerseits leicht Vektorrechenwerke aufbauen, andererseits können auch einfach parallele Operationen durchgeführt werden. Durch die Busverbindungen werden entgegen der herkömmlichen Registerkonzepte Datenabhängigkeiten aufgelöst.

Erfindungsgemäß wurde gemäß einem ersten wesentlichen Aspekt erkannt, daß für einen Compiler für rekonfigurierbare Prozessoren die Konzepte von vektorisierenden Compilern und parallelisierenden (z. B. VLIW) Compilern zugleich anzuwenden sind und somit auf feingranularer Ebene zu vektorisieren und parallelisieren ist.

Ein wesentlicher Vorteil besteht darin, dass der Compiler nicht auf eine fest vorgegebene Hardwarestruktur abbilden muss, sondern die Hardwarestruktur so konfiguriert wird, dass sie optimal für die Abbildung des jeweiligen compilierten Algorithmus geeignet ist.

Beschreibung der erfindungsgemäßen Kompilier- und Datenverarbeitungsvorrichtungsbetriebsverfahren

Moderne Prozessoren weisen zumeist einen Satz an benutzerdefinierbaren Befehlen (UDI) auf, die für Hardwareerweiterungen und/oder spezielle Coprozessoren und Beschleunigern zur Verfügung stehen. Sofern UDIs nicht zur Verfügung stehen, weisen Prozessoren zumindest freie, bislang noch unbenutzte Befehle und/oder Spezialbefeh-

le für Coprozessoren auf - der Einfachheit halber werden alle diese Befehle nachfolgend unter dem Begriff UDI zusammengefasst.

Eine Menge dieser UDIs können nun gemäß einem Aspekt der Erfindung verwendet werden, um eine als Datenpfad in den Prozessor eingekoppelte VPU anzusteuern. Beispielsweise kann durch UDIs das Laden und/oder Löschen und/oder Starten von Konfigurationen ausgelöst werden, und zwar kann eine bestimmte UDI auf eine konstant bleibende und/oder wechselnde Konfiguration Bezug nehmen.

Bevorzugt werden Konfigurationen in einen Konfigurations-cache, der lokal der VPU zugeordnet ist, vorgeladen und/oder in Konfigurationsstacks nach DE 196 51 075.9-53, DE 197 04 728.9 und DE 102 12 621.6-53 vorgeladen, aus denen sie zur Laufzeit bei Auftreten einer eine Konfiguration startende UDI schnell konfiguriert und ausgeführt werden können. Das Konfigurationsvorladen kann in einen mehreren PAEs oder PAs gemeinsamen Konfigurationsmanager und/oder in einen lokalen Konfigurationsspeicher an und/oder in einer PAE erfolgen, wobei dann nur noch die Aktivierung veranlaßt werden muß.

Bevorzugt wird ein Satz an Konfigurationen vorgeladen. Allgemein entspricht bevorzugt je eine Konfiguration einer Lade-UDI. Mit anderen Worten referenzieren die Lade-UDIs auf je eine Konfiguration. Zugleich ist es auch möglich, mit einer Lade-UDI auf eine komplexe Konfigurationsanordnung Bezug zu nehmen, bei der etwa sehr umfangreiche Funktionen, die ein mehrfaches Umlanden des Arrays während der Ausführung erfordern, eine - auch wiederholte - Wave-Rekonfiguration usw. durch eine einzelne UDI referenzierbar sind.

Während des Betriebes können Konfigurationen durch andere ausgewechselt werden und die Lade-UDIs dementsprechend umreferenziert werden. Eine bestimmte Lade-UDI kann also zu einem ersten Zeitpunkt auf eine erste Konfiguration referenzieren und bei einem zweiten Zeitpunkt auf eine mittlerweile neu geladene zweite Konfiguration referenzieren. Es kann dies dadurch geschehen, daß etwa ein Eintrag in einer Referenzliste, auf die gemäß UDI zugegriffen werden soll, geändert wird.

Im Rahmen der Erfindung wird für den Betrieb der VPU ein LOAD/STORE Maschinenmodell zugrundegelegt, wie es beispielsweise von RISC-Prozessoren bekannt ist. Jede Konfiguration wird als ein Befehl verstanden. Separat zu den datenverarbeitenden Konfigurationen sind die LOAD und STORE Konfigurationen.

Ein Datenverarbeitungsablauf (LOAD-PROCESS-STORE) findet dementsprechend z. B. wie folgt statt:

1. LOAD-Konfiguration
Laden der Daten aus z. B. einem externen Speicher, einem ROM eines SOCs, in dem die Gesamtanordnung integriert ist, und/oder der Peripherie in die interne Speicherbank (RAM-PAE, siehe DE 196 54 846.2-53 , DE 100 50 442.6). Die Konfigura-tion umfaßt erforderli-

chenfalls Adressgeneratoren und/oder Zugriffssteuerungen, um Daten von prozessorexternen Speichern und/oder Peripherie zu lesen und in die RAM-PAEs zu schreiben. Die RAM-PAEs können für den Betrieb als mehrdimensionale Datenregister (z. B. Vektorregister) verstanden werden.

2. - (n-1). Datenverarbeitungs-Konfigurationen
Die datenverarbeitenden Konfigurationen werden sequentiell nacheinander in das PA konfiguriert. Die Datenverarbeitung findet entspechend eines LOAD/STORE (RISC) Prozessors bevorzugt ausschliesslich zwischen den RAM-PAEs - die als mehrdimensionale Datenregister verwendet werden - statt.

n. STORE-Konfiguration
Schreiben der Daten aus den internen Speicherbanken (RAM-PAEs) an den externen Speicher bzw. die Peripherie. Die Konfiguration umfaßt Adressgeneratoren und/oder Zugriffssteuerungen, um Daten von den RAM-PAEs an die prozessorexternen Speicher und/oder Peripherie zu schreiben.

Fuer die Grundlagen der LOAD/STORE Operationen sei auf PACT11 verwiesen.

Die Adressgenerierfunktionen der LOAD/STORE-Konfigurationen sind derart optimiert, dass beispielsweise bei einer nicht linearen Zugriffsfolge des Algorithmus auf externe Daten die entsprechenden Adresspattern von den Konfigurationen generiert werden. Die Analyse der Algorithmen und das Erstellen der Adressgeneratoren für LOAD/STORE erfolgt durch den Compiler.

Dieses Arbeitsprinzip kann durch die Abarbeitung von Schleifen einfach verdeutlicht werden. Beispielhaft soll von einer VPU mit 256 Eintraege tiefen RAM-PAEs ausgegangen werden:

Beispiel a):
for i := 1 to 10000
1. LOAD-PROCESS-STORE Zyklus : Lade & verarbeite 1 ... 256
2. LOAD-PROCESS-STORE Zyklus : Lade & verarbeite 257 ... 512
3. LOAD-PROCESS-STORE Zyklus : Lade & verarbeite 513 ... 768
...

Beispiel b):
for i := 1 to 1000
  for j := 1 to 256

1. LOAD-PROCESS-STORE Zyklus : Lade & verarbeite
   i = 1; j = 1 ... 256
2. LOAD-PROCESS-STORE Zyklus : Lade & verarbeite
   i = 2; j = 1 ... 256
3. LOAD-PROCESS-STORE Zyklus : Lade & verarbeite
   i = 3; j = 1 ... 256
...

Beispiel c):

```
for i := 1 to 1000
  for j := 1 to 512

1. LOAD-PROCESS-STORE Zyklus : Lade & verarbeite
     i = 1; j = 1 ... 256
2. LOAD-PROCESS-STORE Zyklus : Lade & verarbeite
     i = 1; j = 257 ... 512
3. LOAD-PROCESS-STORE Zyklus : Lade & verarbeite
     i = 2; j = 1 ... 256
...
```

Von besonderem Vorteil ist, wenn jede Konfiguration als atomar - also nicht unterbrechbar - betrachtet wird. Dadurch wird das Problem gelöst, dass bei einer Unterbrechung die internen Daten des PAs und der interne Status gesichert werden müssen. Während der Ausführung einer Konfiguration wird der jeweilige Status zusammen mit den Daten in die RAM-PAEs geschrieben.

Der Nachteil des Verfahren ist, dass zunächst keine Aussage über das Laufzeitverhalten einer Konfiguration getroffen werden kann. Dadurch entstehen jedoch Nachteile bezüglich der Echtzeitfähigkeit und des Taskwechselverhaltens.

Daher wird als erfindungsgemäß bevorzugt vorgeschlagen, die Laufzeit jeder Konfiguration auf eine bestimmte Maximalanzahl von Takten zu beschränken. Die Laufzeitbeschränkung ist kein wesentlicher Nachteil, da typisch eine Obergrenze bereits durch die Größe der RAM-PAEs und der damit verbundenen Datenmenge festgelegt ist. Sinnigerweise korrespondiert die Größe der RAM-PAEs mit der Maximalanzahl von Datenverarbeitungstakten einer Konfiguration, womit eine typische Konfiguration auf einige 100 bis 1000 Takte beschränkt wird. Durch diese Beschränkung können Multithreading/Hyperthreading-, sowie Realtime-Verfahren zusammen mit einer VPU implementiert sein.

Die Laufzeit von Konfigurationen wird bevorzugt über einen (mit dem Takt oder einem anderen Signal mitlaufenden) Mitlaufzähler bzw. Watchdog, z. B. einen Zähler überwacht. Bei einer Zeitüberschreitung löst der Watchdog einen Interrupt und/oder Trap aus, der ähnlich eines "illegal opcode" Traps von Prozessoren verstanden und behandelt werden kann.

Eine Einschränkung kann zur Reduzierung von Rekonfigurationsvorgängen und zur Performance-Steigerung alternativ eingeführt werden:
Laufende Konfigurationen können den Watchdog retriggern und somit länger ablaufen, ohne gewechselt werden zu müssen. Ein Retrigger ist nur dann zulässig, wenn der Algorithmus einen "sicheren" Zustand (Synchronisationszeitpunkt) erreicht hat, in dem alle Daten und Zustände in die RAM-PAEs geschrieben sind und eine Unterbrechung algorithmisch zulässig ist. Der Nachteil dieser Erweiterung ist, dass eine Konfiguration im Rahmen ihrer Datenverarbeitung in

einen Deadlock laufen könnte, jedoch den Watchdog weiterhin ordentlich retriggert und somit die Konfiguration nicht terminiert.

Eine Blockade der VPU-Ressource durch eine derartige Zombie-Konfiguration kann dadurch verhindert werden, dass das Retriggern des Watchdog durch einen Taskwechsel unterbunden werden kann und somit die Konfiguration zum nächstfolgenden Synchronisationszeitpunkt oder nach einer vorgegebenen Anzahl von Synchronisationszeitpunkten gewechselt wird. Dadurch terminiert zwar der den Zombie aufweisende Task nicht mehr, das Gesamtsystem läuft jedoch ordentlich weiter.

Als eine weitere Methode kann optional Multithreading und/ oder Hyperthreading für das Maschinenmodell bzw. den Prozessor eingeführt werden. Sämtliche VPU-Routinen, also deren Konfigurationen, werden dann bevorzugt als eigener Thread betrachtet. Da die VPU als Rechenwerk in den Prozessor eingekoppelt ist, kann sie als eine Ressource für die Threads betrachtet werden. Der nach den Stand der Technik für Multithreading implementierte Scheduler (siehe auch P 42 21 278.2-09) verteilt automatisch für VPUs programmierte Threads (VPU-Threads) auf diese. Mit anderen Worten verteilt der Scheduler die unterschiedlichen Tasks automatisch innerhalb des Prozessors.

Dadurch entsteht eine weitere Ebene an Parallelismus. Sowohl reine Prozessor-Threads als auch VPU-Threads werden parallel verarbeitet und können automatisch ohne besondere Zusatzmaßnahmen durch den Scheduler verwaltet werden.

Besonders leistungsfähig ist das Verfahren, wenn der Compiler wie bevorzugt und regelmäßig möglich Programme in mehrere parallel abarbeitbare Threads zerlegt und dabei sämtliche VPU-Programmabschnitte in einzelne VPU-Threads aufteilt.

Um einen schnellen Taskwechsel, insbesondere auch Realtime-Systeme zu unterstützen, können mehrere VPU Datenpfade, die jeweils als eigenständige Ressource betrachtet werden, implementiert sein. Gleichzeitig erhöht sich dadurch auch der Grad an Parallelität, da mehrere VPU-Datenpfade parallel nutzbar sind.

Um Realtime-Systeme besonders zu unterstützen, können bestimmte VPU-Ressource für Interrupt-Routinen reserviert sein, sodass für eine Antwort auf einen eintreffenden Interrupt nicht bis zur Terminierung der atomaren, nicht unterbrechbaren Konfigurationen gewartet werden muss. Alternativ dazu können VPU-Ressourcen für Interrupt-Routinen gesperrt sein, d. h. keine Interruptroutine kann eine VPU-Ressource verwenden und/oder einen entsprechenden Thread beinhalten. Damit sind ebenfalls schnelle Interrupt Antwortzeiten gegeben. Da typischerweise innerhalb von Interruptroutinen keine oder nur wenige VPU-performante Algorithmen vorkommen, ist dieses Verfahren bevorzugt. Sofern der Interrupt zu einem Taskwechsel führt, kann währenddessen die VPU-Ressource terminiert werden; im

Rahmen des Taskwechsels steht gewöhnlicherweise ausreichend Zeit zur Verfügung.

Ein bei Taskwechseln auftretendes Problem kann sein, dass der zuvor beschriebene LOAD-PROCESS-STORE Zyklus unterbrochen werden muss, ohne dass sämtliche Daten und/oder Statusinformationen aus den RAM-PAEs in die externen RAMs und/oder Peripheriegeräte geschrieben wurden.

Entsprechend gewöhnlicher Prozessoren (z. B. RISC LOAD/STORE Maschinen) wird nunmehr eine Konfiguration PUSH eingeführt, die, z. B. bei einem Taskwechsel, zwischen den Konfigurationen des LOAD-PROCESS-STORE Zyklus eingefügt werden kann. PUSH sichert die internen Speicherinhalte der RAM-PAEs nach extern, z. B. auf einen Stack; extern bedeutet hier z. B. extern zum PA oder einem PA-Teil, kann aber auch auf Peripherie usw. Bezug nehmen. Insoweit entspricht PUSH somit in seiner Grundlage dem Verfahren klassischer Prozessoren. Nach Ausführung der PUSH Operation kann der Task gewechselt werden, d.h. der aktuelle LOAD-PROCESS-STORE-Zyklus kann abgebrochen werden und ein LOAD-PROCESS-STORE-Zyklus des nächsten Tasks kann ausgeführt werden. Der abgebrochene LOAD-PROCESS-STORE Zyklus wird bei einem nachfolgenden Taskwechsel auf den entsprechenden Task an der Konfiguration (KATS), die nach der letzten durchgeführten Konfiguration folgt, wieder aufgesetzt. Dazu wird vor dem Konfigurieren von KATS eine POP Konfiguration durchgeführt, die wiederum entsprechend der Verfahren bei bekannten Prozessoren die Daten für die RAM-PAEs aus den externen Speichern lädt, z. B. dem Stack.

Als besonders leistungsfähig wurde hierfür eine erweiterte Version der RAM-PAEs nach DE 196 54 595.1-53 und DE 199 26 538.0 erkannt, bei welcher die RAM-PAEs direkten Zugriff auf einen Cache haben (DE 199 26 538.0) (Fall A) oder als besondere Slices innerhalb eines Caches betrachtet werden bzw. direkt gecached werden können (DE 196 54 595.1-53) (Fall B).
Durch den direkten Zugriff der RAM-PAEs auf einen Cache oder die direkte Implementierung der RAM-PAEs in einem Cache können die Speicherinhalte bei einem Taskwechsel schnell und einfach ausgetauscht werden.

Fall A: Die RAM-PAE Inhalte werden über einen bevorzugt separaten und eigenständigen Bus in den Cache geschrieben und aus diesem neu geladen. Die Verwaltung des Caches übernimmt ein Cachekontroller nach dem Stand der Technik. Dabei müssen nur die RAM-PAEs in den Cache geschrieben werden, die gegenüber dem ursprünglichen Inhalt verändert wurden. Dazu kann ein "dirty" Flag für die RAM-PAEs eingeführt werden, welches anzeigt, ob ein RAM-PAE beschrieben und verändert wurde. Daß dafür entsprechende Hardware-Mittel zur Implementierung vorgesehen werden können, sei erwähnt.

Fall B: Die RAM-PAEs liegen direkt im Cache und sind dort als Sonderspeicherstellen markiert, die nicht von den normalen Datentransfers zwischen Prozessor und Speicher beeinflusst werden. Bei

einem Taskwechsel werden andere Cache Abschnitte referenziert. Veränderte RAM-PAEs können mit dirty markiert werden. Die Verwaltung des Caches liegt beim Cachekontroller.

Bei Anwendung der Fälle A und/oder B kann ein Write-Through-Verfahren applikationsabhängig erhebliche Geschwindigkeitsvorteile erzielen. Dabei werden die Daten der RAM-PAEs und/oder Caches bei jedem Schreibzugriff durch die VPU direkt an den externen Speicher durchgeschrieben. Damit bleibt der RAM-PAE und/oder Cacheinhalt gegenüber dem externen Speicher (und/oder Cache) zu jedem Zeitpunkt sauber. Die Notwendigkeit bei einem Taskwechsel die RAM-PAEs gegenüber dem Cache bzw. den Cache gegenüber dem externen Speicher upzudaten, entfällt.

Unter Verwendung derartiger Verfahren können PUSH und POP Konfigurationen entfallen, da die Datentransfers für die Kontext-Switches von der Hardware ausgeführt werden.

Durch die Beschränkung der Laufzeit von Konfigurationen und der Unterstützung schneller Taskwechsel wird die Realtime-Fähigkeit eines VPU-gestützten Prozessors sichergestellt.

Der LOAD-PROZESS-STORE-Zyklus erlaubt eine besonders effiziente Debugging-Methode des Programmcodes nach DE 101 42 904.5. Wenn wie bevorzugt jede Konfiguration als atomar und somit ununterbrechbar betrachtet wird, liegen die für das Debugging relevanten Daten und/oder Zustände grundsätzlich nach Beendigung der Verarbeitung einer Konfiguration in den RAM-PAEs. Der Debugger muss somit lediglich auf die RAM-PAEs zugreifen, um alle wesentlichen Daten und/oder Zustände zu erhalten.

Damit ist die Granularität einer Konfiguration hinreichend debugbar. Sofern Details über die abgearbeiteten Konfigurationen debugged werden müssen, wird nach DE 101 42 904.5 ein Mixed Mode Debugger verwendet, bei welchem die RAM-PAE-Inhalte vor und nach einer Konfiguration gelesen werden und die Konfiguration selbst mittels eines Simulators, der die Abarbeitung der Konfiguration simuliert, überprüft wird.

Sofern die Simulationsergebnisse nicht mit den Speicherinhalten der RAM-PAEs nach Ablauf der auf der VPU verarbeiteten Konfiguration übereinstimmen, ist der Simulator nicht mit der Hardware konsistent und es liegt entweder ein Hardware- oder Simulatorfehler vor, der sodann von dem Hersteller der Hardware bzw. der Simulationssoftware überprüft werden muss.

Es soll besonders darauf hingewiesen werden, dass die Begrenzung der Laufzeit einer Konfiguration auf eine maximale Anzahl an Zyklusn die Anwendung von Mixed-Mode-Debuggern besonders begünstigt, da somit nur eine relativ geringe Anzahl von Zyklusn simuliert werden muss.

Durch das beschriebene Verfahren der atomaren Konfigurationen wird auch das Setzen von Breakpoints vereinfacht, da das Überwachen der Daten nach dem Auftreten einer Breakpoint-Bedingung nur an den RAM-PAEs notwendig ist, so dass nur diese mit Breakpoint-Registern und -Vergleichern ausgestattet werden müssen.

In einer erweiterten Hardware-Variante können die PAEs Sequenzer nach DE 196 51 075.9-53 (Fig. 17, 18, 21) und/oder DE 199 26 538.0 aufweisen, wobei beispielsweise Einträge des Konfigurationsstacks (vgl. DE 197 04 728.9, DE 100 28 397.7, DE 102 12 621.6-53) als Codespeicher für einen Sequenzer verwendet werden.

Es wurde erkannt, dass derartige Sequenzer zumeist sehr schwer durch Compiler beherrschbar und nutzbar sind. Daher werden bevorzugt Pseudocodes für diese Sequenzer zur Verfügung gestellt, auf welche Compiler-generierte Assemblerbefehle abgebildet werden. Beispielsweise ist es ineffizient, Opcodes für Division, Wurzel, Potenzen, geometrische Operationen, Komplexe Mathematik, Fliesskomma-Befehle etc. in Hardware zur Verfügung zu stellen. Daher werden derartige Befehle als mehrzyklische Sequenzer-Routinen implementiert, wobei der Compiler bei Bedarf derartige Makros durch den Assembler instantiiert.

Besonders interessant sind die Sequenzer beispielsweise für Applikationen, in welchen häufig Matrix-Berechnungen durchgeführt werden müssen. In diesen Fällen lassen sich komplette Matrix-Operationen wie beispielsweise eine 2x2 Matrixmultiplikation als Makros zusammenfassen und für die Sequenzer zur Verfügung stellen.

Sofern in einer erweiterten Architekturvariante FPGA-Einheiten in den ALU-PAEs implementiert sind, weist der Compiler folgende Option auf:
Bei Auftreten von logischen Operationen innerhalb des vom Compiler zu uebersetzenden Programmes, z.B. &,|,>>,<< etc. generiert der Compiler eine der Operation entsprechende Logikfunktion fuer die FPGA-Einheiten innerhalb der ALU-PAE. Insoweit der Compiler sicher feststellen kann, dass die Funktion keine zeitlichen Abhaenigkeiten gegenueber ihren Eingangs- und Ausgangsdaten aufweist, kann auf das Einfuegen von Registerstufen nach der Funktion verzichtet werden.
Ist eine zeitliche Unabhaengigkeit nicht sicher feststellbar, werden nach der Funktion in der FPGA-Einheit Register hinzukonfiguriert, die eine Verzoegerung um einen Takt und somit die Synchronisation bewirken.
Bei Einfuegen von Registern wird bei der Konfiguration der generierten Konfiguration auf die VPU wird die Anzahl der eingefuegten Registerstufen per FPGA-Einheit in ein Verzoegerungsregister geschrieben, das die Zustandsmaschine der PAE ansteuert. Dadurch kann die Zustandsmaschine das Verwalten der Handshakeprotokolle an die zusaetzlich auftretende Pipelinestufe anpassen.
Nach eine Reset oder einem Rekonfigurationssignal (z.B. Reconfig) (siehe PACT08, PACT16) werden die FPGA-Einheiten neutral geschaltet, d.h. sie lassen die Eingangsdaten ohne Modifikation an den

Ausgang durch. Somit benoetigen unbenutzte FPGA-Einheiten keinerlei Konfigurationsinformation.

Sämtliche erwähnten PACT Patentanmeldungen sind zu Offenbarungszwecken vollumfänglich eingegliedert.

Beliebige weitere Ausgestaltungen und Kombinationen der erläuterten Erfindungen sind möglich und einem Fachmann offensichtlich.

Titel: Prozessorkopplung

Patentansprüche

1. Verfahren zum Betrieb und/oder der Vorbereitung des Betriebs eines herkömmlichen, insbesondere sequenziellen Prozessors und eines rekonfigurierbaren Feldes von Datenverarbeitungseinheiten, insbesondere eines laufzeitrekonfigurierbaren Feldes von Datenverarbeitungseinheiten,
worin
der herkömmliche Prozessor in einem Satz aus einer Vielzahl von vordefinierten und nichtvordefinierten Befehle definierte Befehle abarbeitet
und Datenverarbeitungseinheitenfeldrekonfigurationen auslöst,
dadurch gekennzeichnet daß
die Datenverarbeitungseinheitenfeldrekonfigurationen bzw. Datenverarbeitungseinheitenfeldteil- und/oder -vorladerekonfigurationen im Ansprechen auf das Auftreten von dem Prozessor nicht vordefinierten Befehlen durch diesen ausgelöst und/oder bewirkt werden.

2. Verfahren nach dem vorhergehenden Anspruch, dadurch gekennzeichnet daß mehrere dem Prozessor nicht vordefinierte, sondern vom Benutzer definierte Befehle vorgesehen sind, wobei auf unterschiedliche vom Benutzer definierte Befehle unterschiedliche Datenverarbeitungseinheitenfeldrekonfigurationen bewirkt werden.

3. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß eine Referenzierung auf Datenverarbeitungseinheitenfeldrekonfigurationen vorgesehen wird, um die Verwaltung der Datenverarbeitungseinheitenfeldrekonfigurationen zu unterstützen und insbesondere den Wechsel der Zuordnung von zu ladenden Konfigurationen zu vom Benutzer definierten Befehle zu erleichtern.

4. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet daß mehrere Konfigurationen simultan geladen, insbesondere für eine auch nur evtl. mögliche und/oder erwartete Ausführung, vorgeladen werden.

5. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß im Befehlssatz der CPU und/oder des herkömmlichen, insbesondere sequenziellen Prozessors load/store-Instruktionen mit integrierter Statusabfrage (load_rdy, store_ack) vorgesehen werden, die insbesondere zur Steuerung von Schreib- und/oder Leseoperationen verwendet werden.

6. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet daß auf der VPU auszuführenden Konfigurationen durch einen Instruktionsdekoder (0105) der CPU und/oder des anderen herkömmlichen, insbesondere sequenziellen Prozessors selektiert werden, wobei dieser Instruktionsdekoder bestimmte, für die VPU bestimmte Instruktionen erkennt und bevorzugt, sofern dort vorhanden, deren Konfigurationseinheit (0106) derart ansteuert, dass diese die entsprechenden Konfigurationen aus einem der CT zugeordneten Speicher (0107), der insbesondere mit der CPU geshared werden oder derselbe wie der Arbeitsspeicher der CPU sein kann, in konfigurierbare Datenverarbeitungseinheitfeld, das insbesondere als Array aus PAEs (PA, 0108) gebildet ist, lädt.

7. Verfahren insbesondere nach dem vorhergehenden Anspruch zum Betrieb und/oder der Vorbereitung des Betriebs eines herkömmlichen, insbesondere sequenziellen Prozessors und eines rekonfigurierbaren Feldes von Datenverarbeitungseinheiten, insbesondere eines laufzeitrekonfigurierbaren Feldes von Datenverarbeitungseinheiten,
worin der herkömmliche, insbesondere sequenziellen Prozessor zumindest zeitweise in einem Multithreadingbetrieb betrieben wird.

8. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß zur Betriebsvorbereitung insbesondere durch einen Compiler eine Anwendung in eine Vielzahl Threads zerlegt wird.

9. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß für den herkömmlichen, insbesondere sequentiellen Prozessors Interruptroutinen insbesondere frei von Code für das rekonfigurierbare Feld von Datenverarbeitungseinheiten vorgesehen wird.

10. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß mehrere VPU Datenpfade implementiert werden, die jeweils als eigenständige Ressource angesprochen werden und/oder parallel genutzt werden.

11. Verfahren nach einem der vorhergehenden Ansprüche, worin auf dem rekonfigurierbaren Feld von Datenverarbeitungseinheiten schwer parallelisierbare Operationen, insbesondere die Bestimmung von Divisionen, Wurzeln, Potenzen, geometrischen Operationen, komplexmathematische Operation und/oder Fliesskomma-Berechnungen verlagert werden und diese in form mehrzyklischer Sequenzer-Routinen auf dem rekonfigurierbaren Feld von Datenverarbeitungseinheiten implementiert werden, insbesondere durch Instantiierung von Makros.

12. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß die Datenzugriffe vor dem Betrieb insbesondere bei der Compilierung derart umsortiert werden, dass eine

verbesserte, bevorzugt weitgehende, insbesondere wenigstens im wesentlichen maximale Unabhängigkeit zwischen den Zugriffen durch den Datenpfad der CPU und der VPU vorliegt, um so Laufzeitunterschiede zwischen CPU-Datenpfad und VPU-Datenpfad auszugleichen, und/oder insbesondere, wenn der Laufzeitunterschied zu groß bleibt, per Compiler NOP-Zyklen eingefügt und/oder per Hardware so lange WarteZyklen im CPU-Datenpfad generiert werden, bis notwendige Daten für eine Weiterverarbeitung von der VPU vorliegen und insbesondere in das Register geschrieben wurden oder dies erwartet werden kann, was insbesondere durch Setzen eines zusätzlichen Bits im Register angezeigt werden kann.

13. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß für den Betrieb des rekonfigurierbaren Feldes von Datenverarbeitungseinheiten LOAD und/oder STORE Konfigurationen vorgesehen werden, wobei insbesondere eine LOAD-Konfiguration derart ausgestaltet ist, daß Daten aus z. B. einem externen Speicher in einen internen Speicher geladen werden, wozu insbesondere Adressgeneratoren und/oder Zugriffssteuerungen konfiguriert werden, um Daten von prozessorexternen Speichern und/oder Peripherie zu lesen und in die internen Speicher, insbesondere RAM-PAEs zu schreiben und zwar insbesondere derart wie beim Betrieb als mehrdimensionale Datenregister (z. B. Vektorregister) und/oder wobei weiter insbesondere Daten aus den internen Speichern (RAM-PAEs) an den externen Speicher bzw. die Peripherie geschrieben werden, wozu insbesondere Adressgeneratoren und/oder Zugriffssteuerungen konfiguriert werden, wobei insbesondere zumindest jeweils teilweise Adressgenerierfunktionen derart optimiert werden, dass bei einer nicht linearen Zugriffsfolge des Algorithmus auf externe Daten die entsprechenden Adresspattern von den Konfigurationen generiert werden.

14. Verfahren nach einem der vorhergehenden Ansprüche, worin zur Betriebsvorbereitung ein Debugging erfolgt, insbesondere unter Verwendung von LOAD und/oder STORE Konfigurationen, insbesondere durch Ausführung eines LOAD-PROZESS-STORE-Zyklus, wobei für das Debugging relevante Daten und/oder Zustände nach Beendigung der Verarbeitung einer Konfiguration in den RAM-PAEs liegen und zum Debugging hierauf zugegriffen wird, wobei insbesondere laufzeitbegrenzte bzw. watchdogüberwachte Konfigurationen oder Konfigurationsatome debuggt werden.

15. Verfahren nach dem vorhergehenden Anspruch, dadurch gekennzeichnet, daß das Verhalten der Anordung zur Betriebsvorbereitung simuliert wird.

16. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß zumindest zeitweise eine PUSH Konfiguration auf das Feld konfiguriert wird, insbesondere bei einem Taskwechsel und/oder zwischen den Konfigurationen des LOAD-PROCESS-STORE Zyklus eingefügt wird und die internen Speiche-

rinhalte der Feldinterenn Speicher, insbesondere von RAM-PAEs nach extern sichert, insbesondere auf einen Stack, wobei bevorzugt nach der Push-Konfigurationsabarbeitung auf einen anderen Task gewechselt wird, d.h. der aktuelle LOAD-PROCESS-STORE-Zyklus kann abgebrochen werden und ein LOAD-PROCESS-STORE-Zyklus des nächsten Tasks kann ausgeführt werden und/oder daß zumindest zeitweise eine POP-Konfiguration auf das Feld konfiguriert wird, um Daten aus den externen Speichern wie einem Stack zu laden,

17. Verfahrens nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß ein Scheduler zur Verwendung von Multithreading und/oder Hyperthreading Technologien vorgesehen wird, der Applikationen und/oder Applikationsteile (Threads) feingranular auf Ressourcen innerhalb des Prozessors verteilt.

18. Verfahren insbesondere nach einem der vorhergehenden Ansprüche zum Betrieb und/oder der Vorbereitung des Betriebs eines herkömmlichen, insbesondere sequenziellen Prozessors und eines rekonfigurierbaren Feldes von Datenverarbeitungseinheiten, insbesondere eines laufzeitrekonfigurierbaren Feldes von Datenverarbeitungseinheiten, dadurch gekennzeichnet, daß eine über die zu ladende Konfiguration als nicht unterbrechbar behandelt und/oder betrachtet wird.

19. Verfahren insbesondere nach einem der vorhergehenden Ansprüche zum Betrieb und/oder der Vorbereitung des Betriebs eines herkömmlichen, insbesondere sequenziellen Prozessors und eines rekonfigurierbaren Feldes von Datenverarbeitungseinheiten, insbesondere eines laufzeitrekonfigurierbaren Feldes von Datenverarbeitungseinheiten, worin der herkömmliche Prozessor in einem Satz aus einer Vielzahl von vordefinierten und nichtvordefinierten Befehle definierte Befehle abarbeitet und Datenverarbeitungseinheitenfeldrekonfigurationen auslöst, dadurch gekennzeichnet, daß jede Konfiguration oder, gleichbedeutend insbesondere bei Vor-Laden einer Vielzahl von Konfigurationsgruppen für Zwecke der alternativen und/oder kurz nacheinander ablaufenden Ausführung, jede Konfigurationsgruppe, auf eine bestimmte Maximalzahl an Laufzeittakten begrenzt wird.

20. Verfahren nach dem vorhergehenden Anspruch, dadurch gekennzeichnet, daß die Maximalzahl insbesondere durch Neu-Antriggern bzw. Rücksetzen eines Watchdog- Mitlaufzählers, konfigurationsseitig erhöhbar ist.

21. Verfahren nach dem vorhergehenden Anspruch, dadurch gekennzeichnet, daß eine per se mögliche konfigurationsseitige Maximalzahlerhöhung unterbindbar ist, insbesondere bei und/oder durch Taskswitching und/oder ein Maximalzahlerhöhungshäufigkeitsmitlaufzähler zur Begrenzung der Anzahl an Malen, die eine Maximalzahlerhöhung durch eine einzelne Konfiguration erfolgt, vorgesehen ist.

22. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß auf das insbesondere durch einen Mitlaufzähler erfaßte tatsächliche oder vermutliche Auftreten einer nicht terminierenden Konfiguration hin ein Prozessorexceptionsignal generiert wird.

23. Verfahren insbesondere nach einem der vorhergehenden Ansprüche zum Betrieb und/oder der Vorbereitung des Betriebs eines herkömmlichen, insbesondere sequenziellen Prozessors und eines rekonfigurierbaren Feldes von Datenverarbeitungseinheiten, insbesondere eines laufzeitrekonfigurierbaren Feldes von Datenverarbeitungseinheiten, dadurch gekennzeichnet, daß eine Laufzeitabschätzung für die Konfigurationsausführung vorgenommen wird, um einen der Laufzeit adäquaten Betrieb des Prozessors zu ermöglichen.

24. Verfahren zum Betrieb und/oder der Vorbereitung des Betriebs eines herkömmlichen, insbesondere sequentiellen Prozessors und eines rekonfigurierbaren Feldes von Datenverarbeitungseinheiten, insbesondere eines laufzeitrekonfigurierbaren Feldes von Datenverarbeitungseinheiten,
worin Daten zwischen Prozessor und Datenverarbeitungseinheitfeld ausgetauscht werden, dadurch gekennzeichnet, daß Daten vom Datenverarbeitungseinheitfeld in einem Prozessorcache abgelegt und/oder von dort erhalten werden.

25. Verfahren nach dem vorhergehenden Anspruch, dadurch gekennzeichnet, daß eine Cachebereichsmarkierung zur Feststellung als "dirty" geltender Cachebereiche vorgesehen wird.

26. Verfahren nach dem vorhergehenden Anspruch, dadurch gekennzeichnet, daß ein insbesondere cachecontrollerbewirktes hiddenwriteback (verborgenes Rückschreiben) insbesondere zum Cachesauberhalten vorgesehen wird.

27. Vorrichtung insbesondere zur Ausführung eines Verfahrens nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß das oder ein zumindest partiell mit rekonfigurierbaren Einheiten gebildetes Prozessorfeld FPGA-artige Schaltkreisbereiche aufweist, insbesondere als separate rekonfigurierbare Einheiten und/oder als ein oder Teil eines Datenpfades zwischen grobgranular rekonfigurierbaren Einheiten und/oder I/O-Anschlußbereichen, insbesondere ALU-Einheiten und/oder als Teil einer zumindest eine ALU-Einheit enthaltenden Prozessorfeldzelle.

28. Vorrichtung nach dem vorhergehenden Anspruch, dadurch gekennzeichnet, daß im Datenpfad zwischen grobgranular rekonfigurierbaren Einheiten und/oder I/O-Anschlußbereichen vorhandene FPGA-artige Schaltkreisbereiche vorgesehen sind, die bei Nichtverwendung und/oder im Resetzustand einen datenunverändernden Durchlauf erlauben.

29. Vorrichtung nach einem der vorhergehenden Vorrichtungsansprüche, dadurch gekennzeichnet, daß zur Verwendung von Multithreading und/oder Hyperthreading Technologien ein hardwareimplementierter Scheduler vorgesehen ist, der dazu ausgebildet ist, feingranular Applikationen und/oder Applikationsteile (Thread) auf Ressourcen innerhalb des Prozessors zu verteilen.

Zusammenfassung

Es wird beschrieben, wie eine Kopplung eines herkömmlichen, insbesondere sequenziellen Prozessors und eines rekonfigurierbaren Feldes von Datenverarbeitungseinheiten, insbesondere eines laufzeitrekonfigurierbaren Feldes von Datenverarbeitungseinheiten ausgestaltbar ist.

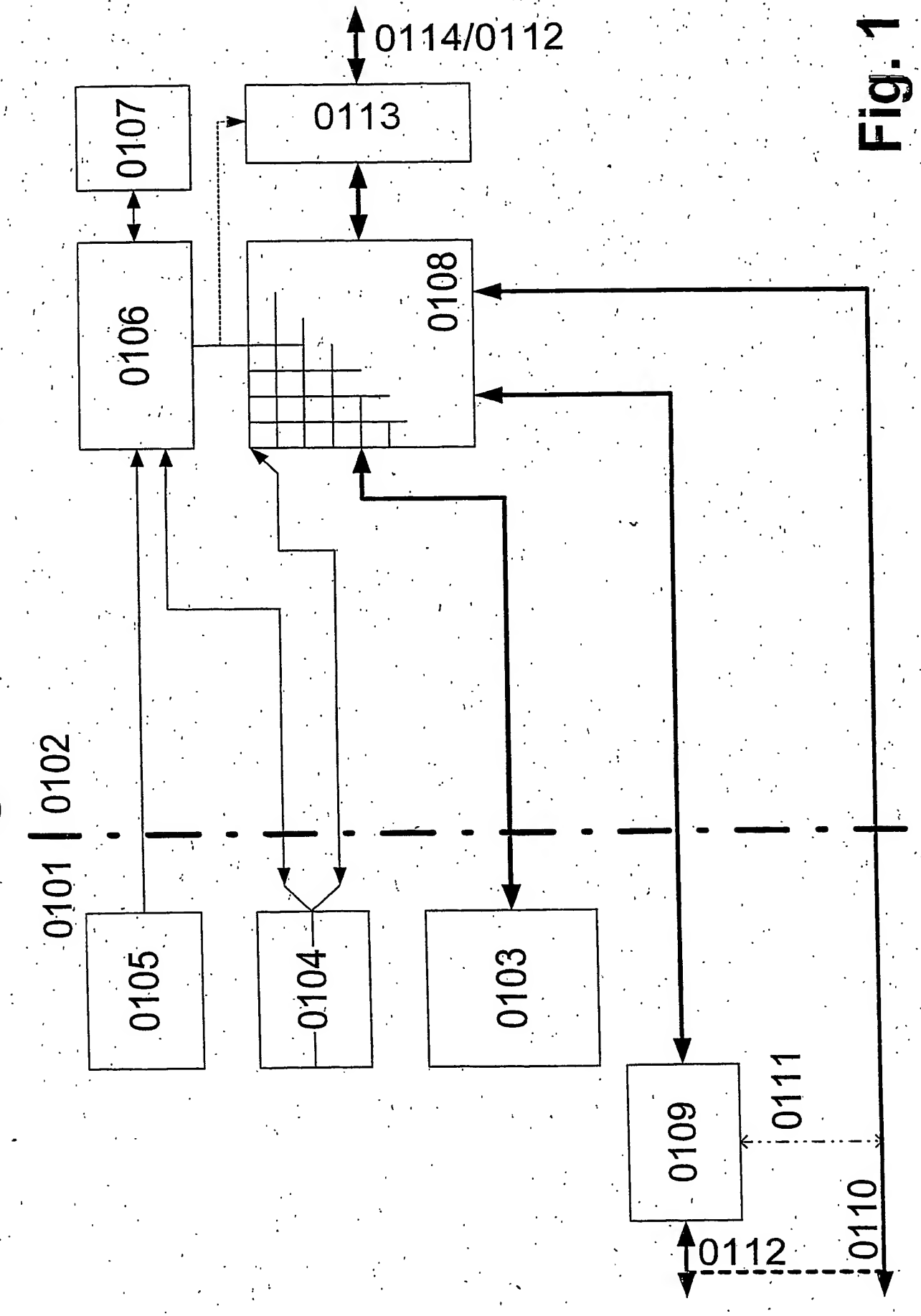

0114/0112
0113
0107
0106
0108
0102
0101
0105
0104
0103
0109
0111
0110
0112

Fig. 1

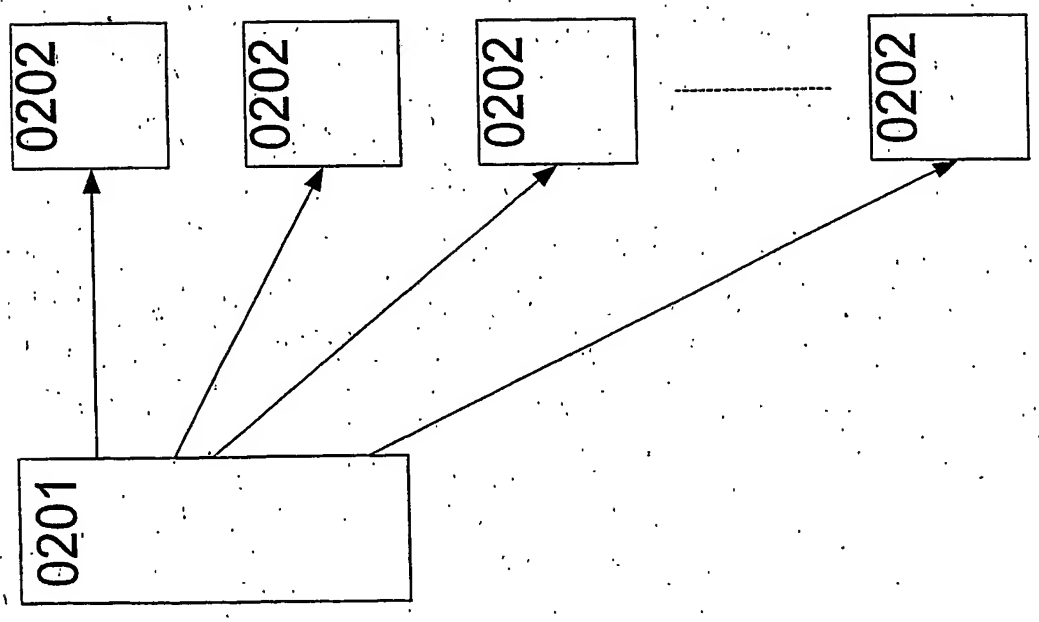


Fig. 2

Fig. 3

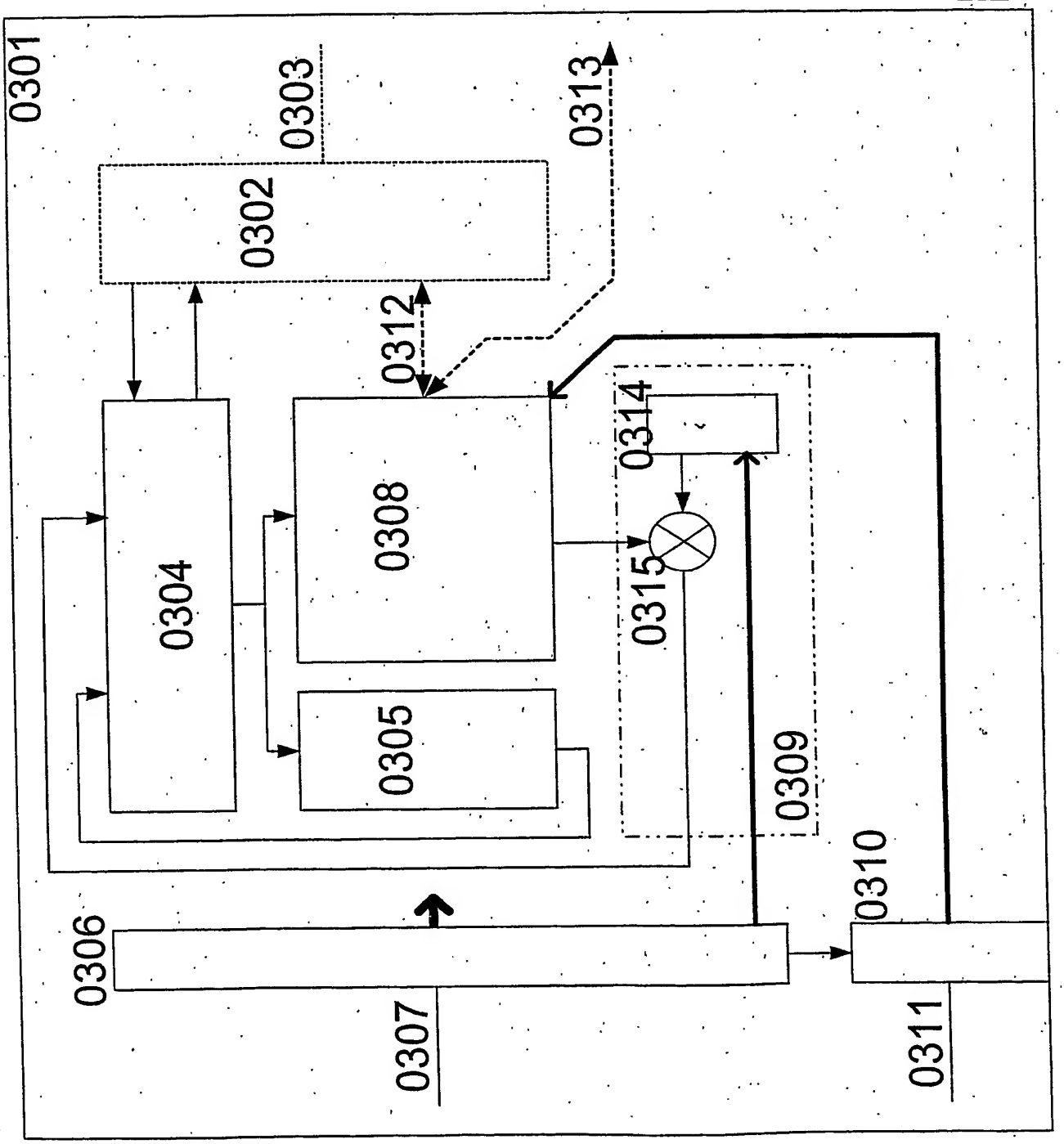
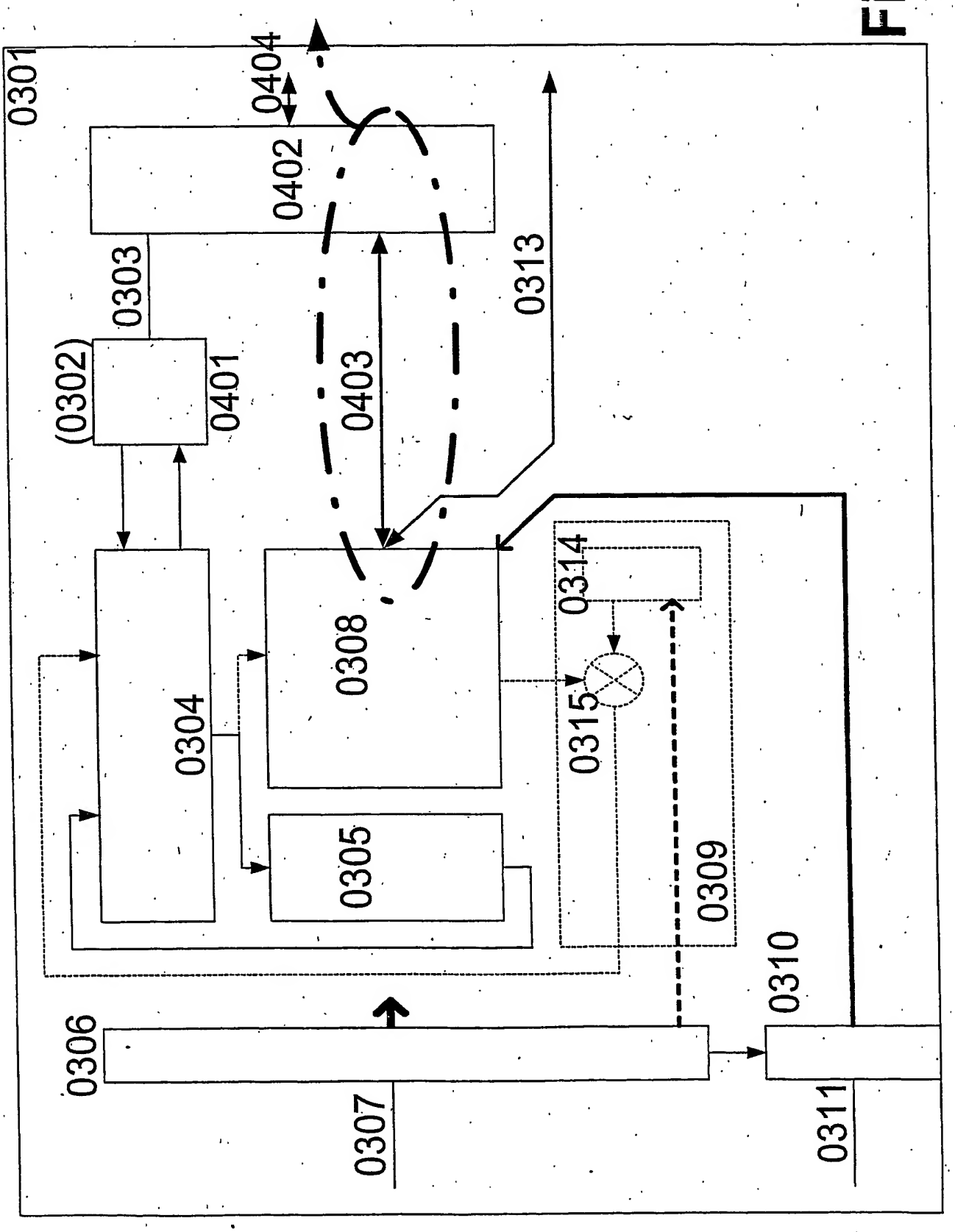

0301
0303
0313
0302
0312
0314
0308
0304
0315
0305
0309
0310
0306
0307
0311

Fig. 4


0301
0404
0402
0303
(0302)
0401
0403
0313
0304
0308
0314
0315
0309
0305
0310
0306
0307
0311

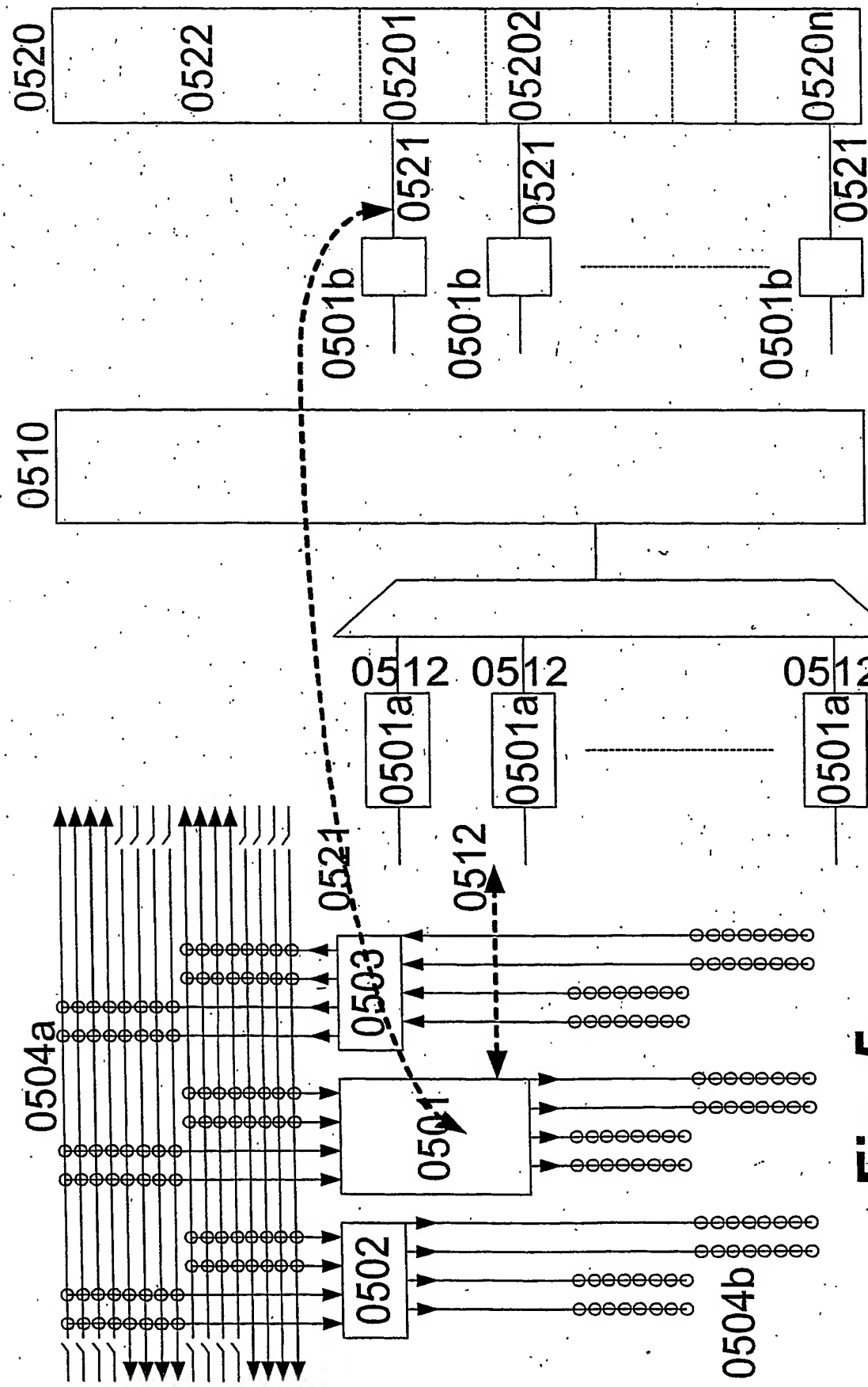


Fig. 5b

Fig. 5a

Fig. 5

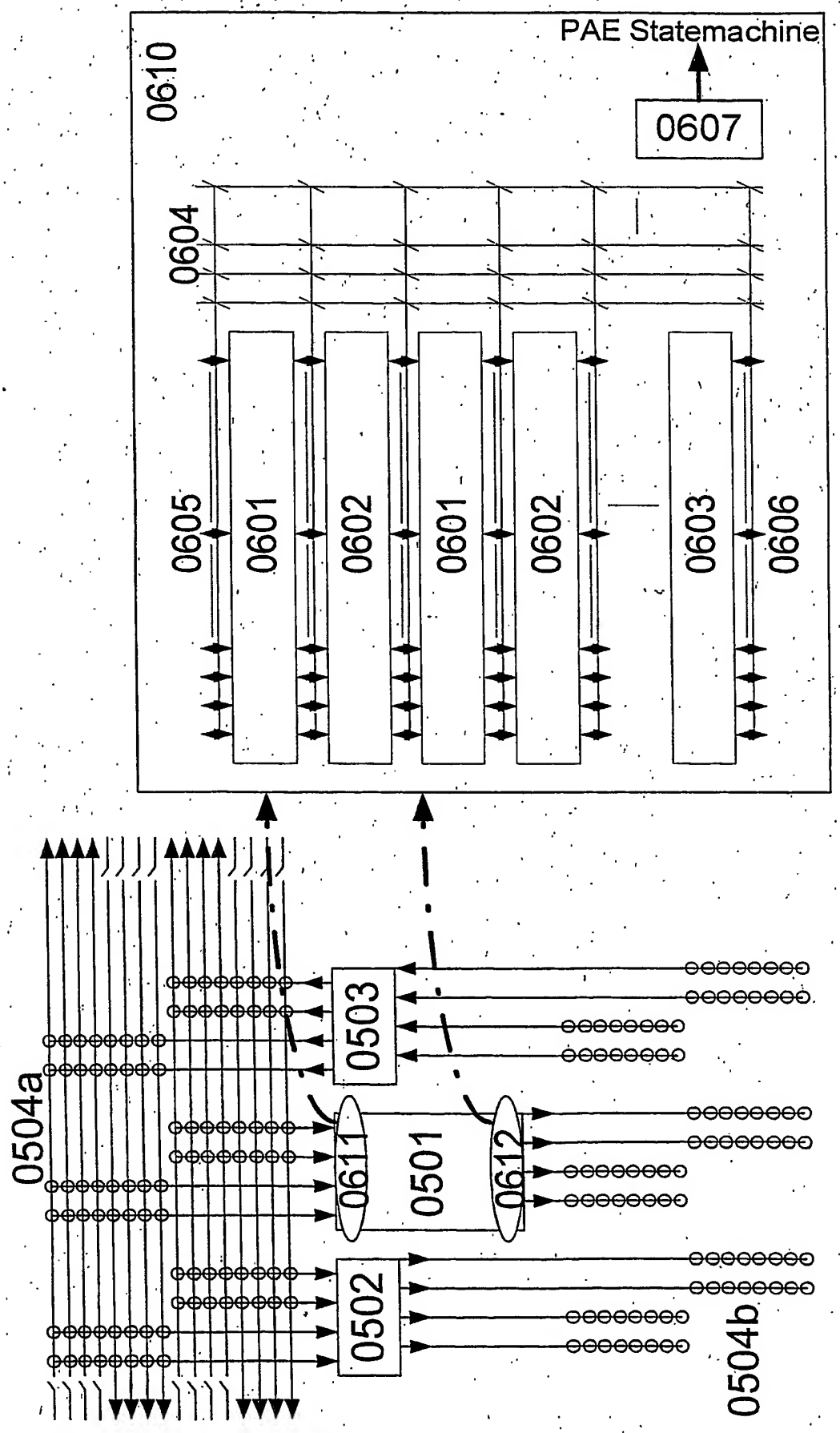
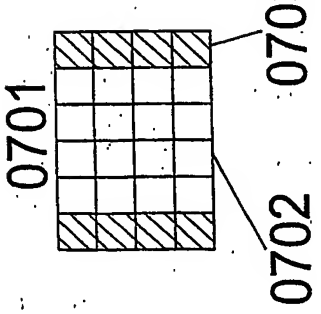

PAE Statemachine
0607
0610
0604
0605
0601
0602
0601
0602
0603
0606
0504a
0503
0611
0501
0612
0502
0504b

Fig. 6



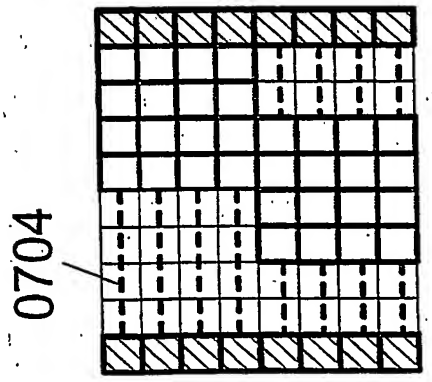


Fig. 7a

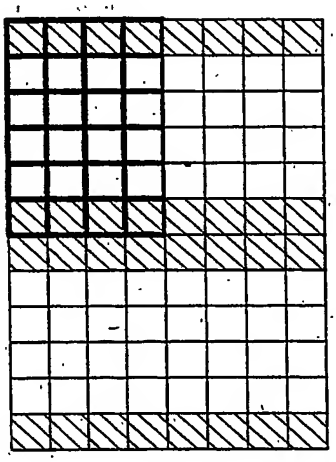


Fig. 7b

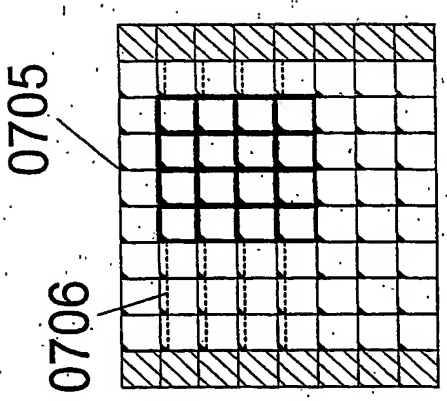


Fig. 7c